2006

Master's project website: A comparison of several server-side technologies

Prabhakaran Nagarajan

Master's Project Website Implementation:
A comparison of several server-side technologies

Graduate Project Report for MS Degree in Computer Science

By
Prabhakaran Nagarajan
May 12, 2004

Committee:
Chairman: Dr. Axel T. Schreiner
Reader: Dr. James E. Heliotis
Observer: Alan Kaminsky

Department of Computer Science
Rochester Institute of Technology

Table of Contents

1 Abstract

1.1 Motivation

Every Graduate student in the Computer Science department at RIT has to complete either a Thesis or Project as part of their degree requirements. The project or thesis completion process involves creation of multiple documents; the Proposal, Final Report and Defense presentation being the mandatory ones. Additional documents and links like User guides, Help documentation, references used etc. are optional and may be submitted as part of the final documentation.

Having a common repository that allows students to browse project / thesis work already undertaken by RIT alumni will therefore help in multiple ways. Students looking for ideas can browse the repository and find areas that can be enhanced. Or, they can look at the projects chaired by the faculty members, and get an insight into the research interests of the faculty, and can approach the faculty members for new ideas. Even students who have a project concept in mind can browse through the repository to get a better understanding on how to organize and write their proposal or report.

With the World Wide Web being ubiquitous these days, the most accessible way of browsing the repository would be if it were available on the web. These days, there are multiple technologies available to publish information on the web, each having its own strengths and weaknesses. Having numerous approaches to solve the same problem poses an interesting question: What is the optimal/recommended approach to solving the problem? What web technologies possess the features required to implement the desired functionality, and at the same time are easy to use, and provide satisfactory response times?

One of these enabling technologies is java servlets, which was used by Dr. Axel Schreiner to design the existing Master's project website. While this is one possible implementation, some competing and some complementing technologies exist that allow for equally elegant solutions. This project aims to select two such technologies, and re-implement the Masters website using the two.

1.2 Existing Solution

The current Masters project web site was designed and implemented by Dr. Axel Schreiner using java servlets. The implementation consists of two sites, a static site and an admin site. The static site contains generated html – pages containing project details, as well as pages that group the projects by faculty, year or student name. The admin site is servlet based, with functionality to create project and manage files. There is no persistent data store; the information is stored as XML, validated against the ms.dtd document type definition. The HTML for the static site is generated by from XML by periodic cron jobs. Security is based on roles. Users belonging to the "advisor" role can edit documents under their sub-tree in the "static" directory. Users in the "supervisor" role can edit any document under the "static" directory (no sub-tree restrictions). This allows an admin-type user to edit any project information without having to know the faculty member's account details.

1.3 Goal

The goal of this project was to select two web-enabling technologies that were either extensively used, and/or were defining technologies of the future, with the aim of comparing the technologies used to implement them. The implementations were also compared against Dr. Axel Schreiner's solution. The comparison focused on the ease of learning/using the new technology, the flexibility of the API, as well as the perceived performance. This served the dual purpose of comparing the three technologies involved, as well as allowing students to browse the Masters Project website for useful information.

The two implementations for this project were done using (1) PHP/MySQL and (2) Web Services/ Web Forms using ASP.NET.

1.4 Composition

This project consists of two parts:
The first is an implementation based on PHP and MySQL. One of the most commonly used server-side scripting technologies today is **PHP.** PHP, a recursive acronym for **PHP: Hypertext Preprocessor**, is an open source initiative headed by the Apache Software Foundation, and is one of the more widely adopted scripting languages. PHP by itself provides for dynamic page creation, and is available on multiple platforms. **MySQL** is an open source relational database widely used in conjunction with PHP. Considering that the CS department at RIT is in the process of migrating the department web site to PHP and MySQL, choosing PHP as one of the implementations serves the dual purpose of comparing a popular scripting technology, as well as integrating it with the rest of the CS web site.

The second implementation is based on **ASP.NET** technologies. Web Services are fundamental building blocks in the move towards distributed computing on the Internet. The primary advantage of the web service architecture is that it allows programs written in different languages and on different platforms to communicate with each other in a standards-based way. Web Services are significantly less complex than earlier approaches that made the same promises, and as an additional advantage, work with standard protocols – XML, HTTP and TCP/IP. ASP.NET makes building XML Web Services very easy. The infrastructure provided by ASP.NET allows for building web services conforming to industry standards such as SOAP, XML and WSDL, while leveraging ASP.NET's performance, state management and authentication functions. This project uses XML Web Services created using ASP.NET as the second server side technology. The ASP.NET solution can only be hosted on a Windows platform, and for this project, the ASP.NET development environment's (Visual Studio) built-in web server was used. The client for the Web Service is also ASP.NET based - Web Forms. Web Forms present information to the user in any browser (or client device) and implements application logic using server-side controls. The output may contain almost any HTTP-capable language, including HTML, XML, WML and ECMAScript. For this project, the output is HTML, with the client visual interface modeled as closely as possible to the PHP UI implementation. Emphasis is placed on the ease of web service creation, and the complexity involved in a client-side invocation.

The implementations are compared against each other as well as the existing servlet based website authored by Dr. Schreiner.

2 System Analysis

This section briefly describes the functional aspects of the Masters project website. Both implementations attempt to provide the same functionality currently available with Dr. Axel Schreiner's implementation. The current functionality is described using use cases, and will be cross-referenced in the design sections. The existing implementation can be broadly divided into two sections - the admin section, and the "student" or "visitor" section. The following two diagrams depict the various use cases:

2.1 Browse Projects

The system should allow users (both faculty members and students, collectively referred to as visitors) to browse the project website by year the project was initiated, by the advisor/observer/reader for a project, alphabetically by project title and alphabetically by graduate student name. Visitors need not be authenticated to be able to browse the site. Visitors can also search for a particular project from the website. The search can be based on keywords for the project, or can be part of the project title.

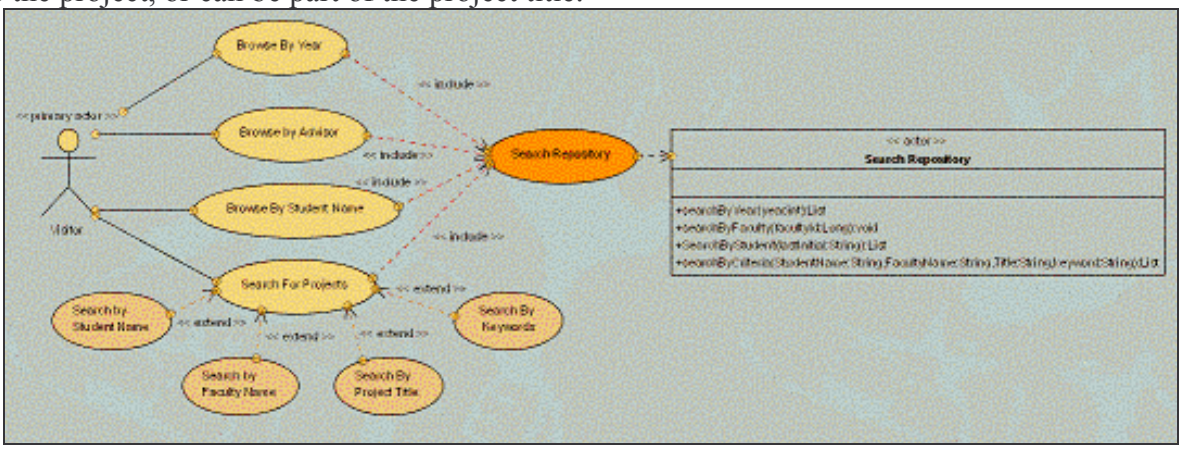

Figure 1: Browse Projects Use Cases

Primary Actor:	Any visitor to the site
Goal in Context:	Visitor visits the site to browse through the list of projects undertaken by Masters students at RIT
Scope:	System - the Masters project website.
Level:	User
Stakeholders:	Visitor - wants to browse the site
Interests:	Visitor - should be able to browse projects started on a given year Visitor - should be able to browse projects advised by selected faculty Visitor - should be able to browse for projects done by selected student Visitor - should be able to search for projects based on keywords Visitor - should be able to search for projects based on project title Visitor - should be able to search for projects advised by faculty name Visitor - should be able to search for projects undertaken by student.
Precondition:	None
Trigger:	None
Main Success Scenario:	1. Visitor finds the project he/she was searching for 2. System displays all the project details.

2.2 Manage Projects

To be able to access the edit features of the site, faculty members must sign in. Once logged in, the committee chairman can create or edit project details for students they currently advise. The edit page allows for entering filter criteria to narrow the search of projects to view/edit. The faculty can edit only those projects for which (s) he acts as the committee chairman; however, an administrator (Graduate Coordinator) can edit project details for all projects. The edit page also allows for the faculty to create a new project – by clicking on the "Create New" button. Creating a new project automatically assigns the logged in faculty as the committee chairman.

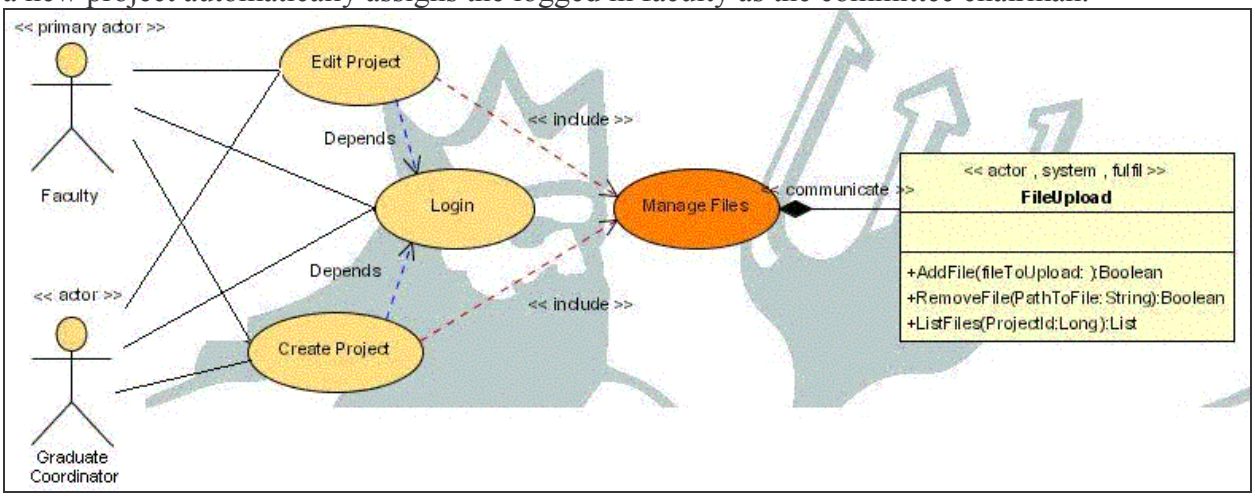

Figure 2: Manage Projects Use Cases

Primary Actor:	Faculty Member
Actor:	Graduate Coordinator or Administrator
Goal in Context:	Project Chairman creates/edits projects done by students. Graduate Coordinator can edit project chaired by any faculty.
Scope:	System - the Masters project website.
Level:	User
Stakeholders:	Faculty - wants to create / edit a new project Graduate Coordinator – wants to edit projects undertaken by any graduate student in addition to creating new projects.
Interests:	Faculty - should be able to create a new project, system should default the project chairman to the logged in faculty. Faculty - should be able to edit projects they chaired. Faculty - should be able to add/delete files for any projects they chaired. Administrator – should be able to edit any project. Administrator – should be able to add/delete project files for any project. Visitor - should be able to view project details
Precondition:	Faculty must be logged in
Trigger:	A student approaches faculty with a project idea, and submits a proposal for the same.
Main Success Scenario:	1. Faculty member creates/edits projects 2. System stores the project details successfully 3. System retrieves and displays project information successfully to any user browsing the site.

Note: The actors Administrator and Graduate Coordinator are used interchangeably in this document.

3 Architectural Overview

This section describes in brief detail the various software / technologies that were used in the two implementations, and how they interact with each other. This includes the technology itself, the development platform, and the **I**ntegrated **D**evelopment **E**nvironment used.

The development machine used is an AMD Athlon XP 1800+ 1.53GHz machine, with 512Mb RAM, running Windows XP Professional edition. ZoneAlarm is the personal firewall used.

3.1 PHP/MySQL Implementation

3.1.1 Technology Overview: Apache (Web Server)

Apache has been the most popular web server on the Internet since its inception in 1996, with over 64% of all web sites on the Internet using Apache to serve web pages. The Apache project is a collaborative software development effort aimed at creating a robust, commercial-grade, feature rich, and freely available source code implementation of a HTTP (web) server[1]. Apache has two code bases, version 1.3.x and 2.0.x. The 2.0.x has significant API changes when compared to the 1.3 versions, but not all modules are available in 2.0. Since some 2.0 modules are still under development, and Apache 2.0 has issues with some firewalls (ZoneAlarm in particular), this project used Apache **1.3.28** as the development web server. At the time of deployment, if version 2.0 is stable, and all required modules available, the deployment server could be the latest stable version of Apache (2.0.48).

3.1.2 Technology Overview: PHP (Scripting Language)

PHP is a HTML-embedded scripting language. It is an open-source, cross-platform, web-centric, server-side scripting language with the fastest adoption rate for developing dynamic web pages. PHP gets installed on top of the web server software. It works with versions of Apache, Microsoft IIS, Netscape Enterprise Server and other server software packages. PHP is available in all flavors of *nix, Windows and Mac OS X. A PHP page consists of HTML code containing embedded PHP code. When the browser makes a request to a PHP page, the web server defers processing of the page to the PHP server. The PHP server interprets and processes the PHP code. The resulting HTML is sent to the browser and displayed to the user. For this project, **PHP 4.3.3** was used with **Apache 1.3.28.**

3.1.3 Technology Overview: MySQL (Database)

MySQL, the most popular Open Source database, is a very fast, multi-threaded, multi-user and robust relational database server with a SQL front end. MySQL was designed to work with small and mid-sized databases, and the MySQL team has been mainly concentrating on speed and reliability. The feature set currently supported by MySQL is sufficient for heavy web/logging usage and mission-critical usage. The latest stable version of MySQL is **4.1**, which is currently SQL-92 and ODBC 3.51 compliant was used for **both** implementations.

3.1.4 Software Overview: PHPEdit (IDE)

PHPEdit is one of the best IDE under Windows to work with PHP. In addition to being free, this IDE offers features like Syntax Highlighting, Code Insight and an Integrated PHP debugger. PHPEdit v. 0.8.0.25 is a stable developmental release that was used in this project.

3.1.5 Software Overview: Wink (Help Tutorial Creator)

Wink is a Tutorial and Presentation creation software, primarily aimed at creating tutorials on how to use software. Distributed as Freeware, using Wink you can capture screenshots of your software, use images that you already have, type-in explanations for each step, create a navigation sequence complete with buttons, delays, titles etc and create a highly effective flash tutorial. Wink 1.0 was used to generate the help files for **both** implementations.

3.2 ASP.NET Implementation

The Microsoft .NET Framework is a multi-language platform for building, deploying, and running Web Services and applications. ASP.NET is a compiled, .NET-based environment, allowing authors to write applications in any .NET compatible language, including Visual Basic .NET, C#, and JScript .NET. For this project, C# was the language used.

3.2.1 Technology Overview: ASP.NET Web Services

The ASP.NET framework simplifies creation of web services. A XML web service developed using ASP.NET automatically supports client communications through the HTTP-GET, HTTP-POST and SOAP protocols. Since HTTP-GET and HTTP-POST pass parameters through url-encoded name/value pairs, the data type support is not as rich as would be through SOAP. Normally, the complex data types are defined using XML Schema Definition (XSD) schema, however, developers using ASP.NET for creating web services do not have to explicitly define XSD schemas. Rather, they can build a managed class. ASP.NET handles mapping class definitions to a XSD schema and mapping object instances to XML data in order to pass it back and forth across the network.

3.2.2 Technology Overview: ASP.NET Web Application

Web Forms is an ASP.NET feature that can be used to create the user interface for Web applications. The user interface programming is divided into two distinct pieces: the visual component and the logic.

The visual element is referred to as the Web Forms *page*. The page consists of a file containing static HTML, or ASP.NET server controls, or both simultaneously. The logic for the Web Forms page consists of code that you create to interact with the form. The programming logic resides in a separate file from the user interface file. This file is referred to as the "code-behind" file and has an ".aspx.vb" or ".aspx.cs" extension. The logic written in the code-behind file can be written in Visual Basic or C#. For this project, C# was used.

3.2.3 Software Overview: "Whidbey" Visual Studio .NET (IDE)

Visual Studio .NET provides the tools needed to design, develop, debug, and deploy Web applications, XML Web services, and traditional client applications. The latest Visual Studio .NET offering from Microsoft, codenamed "Whidbey" enables rapid application development by providing better source code editing tools, richer Visual Designers, better web projects support (has its own built-in ASP.NET web server, no longer dependent on IIS or FrontPage extensions to be installed), support for pre-compiling ASP.NET applications and powerful Object Data binding.

Since the "Whidbey" release of Visual Studio and the .NET framework delivers a new set of tools and functionality to simplify application development without sacrificing existing language functionality, this project used "Whidbey" as the IDE for creating ASP.NET web services and Web Forms.

3.2.4 Software Overview: Poseidon Community Edition

The Community Edition is fully functional UML modeling tool. Fully implemented in Java, it is platform independent and supports all 9 diagrams of the UML. The UML diagrams can be saved in the UML 2.0 Diagram Interchange Standard, as well exported to common graphics formats of gif, jpg and png. The Poseidon Community Edition UML modeling tool was used to model **both implementations.**

4 High Level Design

This section describes briefly the steps involved in setting up the technologies, highlighting the complexities involved in creating a development environment for both implementations. This includes the process of setting up the database server and designing the database; setting up the web server and the PHP scripting engine; configuration of Visual Studio to enable it to act as the web server listening on multiple ports and finally, the steps involved in enabling both the PHP engine and Visual Studio to connect to MySQL.

4.1 MySQL Database Design

Both implementations use MySQL as the data store. MySQL is a Relational Database Management System that stores all the data in tables. The relationships between the tables are maintained through "foreign keys", which essentially link a column in one table with a column in another. The Primary Keys for a table are defined in Red, and the mandatory fields are defined in **bold**. Master tables are defined to preserve data integrity.

4.1.1 Master Tables

CommitteePosition:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
CommitteePositionId	TinyInt (3)	Short	Can be one of the following: 1 Chairman 2 Reader 3 Observer
Description	Varchar (100)	String	Description of the committee PostionId

Table A: Committee Position Master Table

ProjectType:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
ProjectTypeId	TinyInt (3)	Short	Can be one of the following: 1 Project 2 Thesis
Description	Varchar (100)	String	Description of the Project Type

Table B: Project Type Master Table

ReferenceType:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
ReferenceTypeId	TinyInt (3)	Short	Can be one of the following: 1 Project Proposal 2 Project Defense 3 Other Downloads 4 Reference Links
Description	Varchar (100)	String	Description of the Reference Type

Table C: Reference Type Master Table

Role:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
RoleId	TinyInt (3)	Short	Can be one of the following: 1 Visitor 2 Advisor 3 Graduate Coordinator
Description	Varchar (100)	String	Description of the Role

Table D: Role Master Table

Note: The role of Graduate Coordinator is similar to an administrator – an administrator can edit all projects immaterial of who acts as the chairman for the project. The Advisor role implicitly contains the privileges provided by the Visitor role, and the Graduate Coordinator role implicitly grants the Advisor privileges to the person in that role.

4.1.2 Faculty Information

One of the objectives of this project is to have the php version integrated with the existing CS Department website, currently under development. In view of this, the structure of the faculty table will be used from the existing department schema, with some additional columns added. The final database structure for the faculty information is outlined in Table 5.

Faculty:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
FacultyId	Int (10)	Long	Auto increment field, uniquely identifies a faculty member
Username	Varchar (8)	String	Current username used to login to the PHP site, will be used for both implementations
Password	Varchar (10)	String	Plain text password.
Email	Varchar (100)	String	Email address of faculty member. Forgotten/lost passwords will be sent to this email address.
HomePageUrl	Varchar (100)	String	Home page of faculty member. Both implementations do NOT use this field currently; this is a vestigial field from the existing structure.
Salutation	Char (3)	String	Salutation of faculty member, Dr., Mrs. etc.
FirstName	Varchar (100)	String	First Name
LastName	Varchar (100)	String	Last Name
MiddleInitial	Char (1)	String	
SessionId	Varchar (100)	String	Used ONLY in the .NET implementation. Once a faculty member is logged in, subsequent requests have to submit a sessionId, which is validated against this column. This eliminates a need for submitting a username and password for every request.
RoleId	TinyInt (3)	Short	The Graduate Coordinator has a value of 3; all other faculty members have a role value of 2 (default value). The RoleId links the faculty table to the Role table.

Table E : Faculty Information table

4.1.3 Student Information

The student information is captured in the student table, whose structure is shown in Table 6.

Student:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
StudentId	Int (10) unsigned	Long	Auto increment field, uniquely identifies a student
HomePageUrl	Varchar (100)	String	Home page of student. Both implementations do NOT use this field currently, but can be used to link the project details to a student home page/project page.
FirstName	Varchar (100)	String	First Name
LastName	Varchar (100)	String	Last Name
MiddleInitial	Char (1)	String	
RoleId	TinyInt (3)	Short	Currently, all students have a roleId of 1 (default value) - a visitor. This column is present in the table as a potential future enhancement - future versions of this project could allow students to edit their own project details, and having a role in this table would support that. This column links the student table with the Role table.

Table F: Student Information Table

4.1.4 Project Details

The project details are stored in the project table. Database design rules indicate that when a table has multiple fields that hold very similar information in them, they should be represented as rows in a new table, and referentially linked to the parent table. For instance, each project can have up to three faculty members in the committee, in various capacities - chairman, reader and observer. Even though their functions are different, and from a data perspective, the values are different, database design rules indicate that this information should be stored in a new table. Some mapping tables have been defined in addition to the project table that is combined to hold the project information. The structure of the project table and the mapping tables are given below:

Project:			
Column Name	MySQL DataType	.NET Datatype	Column Value
ProjectId	Int (10) unsigned	Long	Auto increment field, uniquely identifies a project
Title	Varchar (100)	String	The title of the project
Year	Year (4)	int	The year the project was started
Quarter	Tinyint (3)	Short	A number denoting quarter the project was started - 1 Fall 2 Winter 3 Spring 4 Summer
TypeId	Tinyint (3)	Short	1 or 2 denoting a project or thesis. Links to ProjectType table
Abstract	Text	String	A short summary of the project/thesis.
StudentId	Int (10) unsigned	Long	Uniquely identifies the student who is working on this project. This links to the Student table.
DefenseDate	Date	Timestamp	The date when the student is scheduled to defend.

Table G: Project Table

ProjectFacultyMap:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
ProjectId	Int (10) unsigned	Long	Links to the project table, identifies the project for which the faculty information is stored
FacultyId	Int (10) unsigned	Long	Links to Faculty table, denotes the faculty associated with the project in the committee position specified.
CommitteePositionId	Tinyint (3)	Short	Links to the CommitteePosition table, denotes the position of the faculty in this project.

Table H: Project Faculty Map Table

ProjectKeywords:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
ProjectId	Int (10) unsigned	Long	Links to the project table, identifies the project for which the faculty information is stored
Keyword	Varchar (100)	String	Keyword associated with the project. Can be used to search for a project.

Table I: Project Keywords Table

ProjectReferences:			
Column Name	**MySQL DataType**	**.NET Datatype**	**Column Value**
ProjectId	Int (10) unsigned	Long	Links to the project table, identifies the project for which the faculty information is stored
ReferenceTypeId	TinyInt (3)	Short	Links to the reference type table.
Url	Varchar (100)	String	The url of any references used in the project.

Table J: Project References Table

4.2 PHP Site Setup

This section briefly describes the actions taken to setup the web server, and the steps for installing the PHP scripting engine as a web server module. The installation and configuration steps for setting up MySQL and integrating are also listed in this section.

For this project, Apache was used as the web server. PHP gets installed on top of the web server software. When the browser makes a request to a PHP page, the web server defers processing of the page to the PHP server. The PHP server interprets and processes the PHP code. The resulting HTML is sent to the browser and displayed to the user. The steps involved in setting up a web server, installing PHP as a module, and MySQL are described below (the installation instructions pertain to the Windows version, which was the development environment):

4.2.1 Apache Server Installation

Apache is open-source software and can either be downloaded with the source code or only the binary distribution without the source code. The binary version of the apache server for Windows can be downloaded from http://mirrors.isc.org/pub/apache/httpd/binaries/win32/apache_1.3.29-win32-x86-no_src.exe.

The following Apache configuration instructions are specific to Windows:

- Apache can either be installed as a service, or started separately as a console application. For this project, apache was installed as a console application. The default installation serves up files from a subdirectory of the installation directory, called htdocs. This can be changed by editing the httpd.conf file and changing the document Root by setting the parameter: **DocumentRoot "*<path to web root>*"**
- Because Apache for Windows is multithreaded, it does not use a separate process for each request, as Apache does with Unix. Instead there are usually only two Apache processes running: a parent process, and a child that handles the requests. Within the child a separate thread handles each request. The number of threads per child is configurable by changing the ThreadsPerChild parameter, the default value being 50.
- The directives that accept filenames as arguments must use Windows filenames instead of Unix ones. However, because Apache uses Unix-style names internally, forward slashes, must be used, not backslashes. Drive letters can be used; if omitted, the drive with the Apache executable will be assumed.
- Apache for Windows has the ability to load modules at runtime, without recompiling the server. To activate dynamic modules, the LoadModule directive must be used. For this project, the PHP module was loaded as a dynamic module.

4.2.2 PHP Engine Installation

PHP, like Apache, is also open-source and can be downloaded as a binary or with the source code. However, unlike Apache, binaries are available only for Windows because Unix/Linux distributions these days come with PHP. Version 4 is the stable version of PHP, while PHP5 is in the beta version. PHP has a direct module interface (SAPI) to Apache 1.3.x, with only experimental support for Apache2. This is one of the reasons for choosing Apache 1.3.x as the web server. Support for MySQL is built in to PHP.

There are two ways to set up PHP to work with Apache 1.3.x on Windows. One is to use the CGI binary (php.exe), the other is to use the Apache module DLL. For this project, PHP version 4 was used, and was setup as a module on Apache. The following steps achieved this:

- Copy the php4ts.dll to the Windows/system32 directory (on WindowsXP)
- Edit the httpd.conf file, and add a LoadModule directive to load the PHP Engine as a dynamic module. This can be done by the following line:
  LoadModule php4_module "*<path to php4apache.dll>*"
- After the ClearModuleList directive, add an AddModule directive to specify to apache to add the PHP module to the list of modules to load.
  AddModule mod_php4.c
- Add an "AddType" directive after the AddModule directive to instruct Apache to redirect all *.php requests to the PHP engine. (In case we want to support other extensions like *.php4, or even *.html, it can also be done).
  AddType application/x-httpd-php .php
- Move the php.ini file to the windows directory from the php installation directory to the c:\windows directory (in WindowsXP)
- The following changes need to be made to the php.ini file:
    - Set the doc_root to the same document root as set in the Apache configuration
    - Set the include_path to all directories (even if they are subdirectories) that might contain any include files or class files that are used by any .php page.
    - Set the SMTP and sendmail_from settings to the SMTP server and the email address to use when sending emails. (This is a windows only setting)
- Set the [Session] settings: Even though session support is enabled by default, PHP supports sessions through files and cookies. The session information is stored in files, and the files are garbage collected after a certain interval. All data related to a particular session was stored in a file in the directory specified by the session.save_path. This should point to a valid directory. The files are deleted by garbage collection (gc) after gc_maxlifetime seconds and with a probability of session.gc_probability % each time session_start() is called.

4.2.3 MySQL Installation

MySQL is an open-source GPL database management system that can be downloaded from www.mysql.com. While there are newer developmental releases of MySQL (version 5 and 4.1), the current stable/production quality release is 4.0. The binary distribution of MySQL was used for the installation in this project. The installation comes with four sample configuration settings: my-huge, my-large, my-medium and my-small, which can be used as templates for defining configuration settings. For this project, using my-small as the template was adequate. The following parameters were changed:

- Set the basedir and datadir to the location where MySQL was installed and the location where you want the database to reside.
- Copy the my.ini (or my.cnf) file to the windows directory
- Choose the windows binary to be used. This project uses mysqld-nt in the development environment. The options are:

- mysqld - Compiled with full debugging and automatic memory allocation checking, symbolic links, and InnoDB and BDB tables.
- mysqld-opt - Optimized binary.
- mysqld-nt - Optimized binary for NT/2000/XP with support for named pipes.
- mysqld-max - Optimized binary with support for symbolic links, InnoDB and BDB
- mysqld-max-nt - as above, but optimized for NT.

The recommended approach to starting MySQL in windows is as a service. Executing the following command can do this:
C:\mysql\bin\mysqld-nt –install

4.2.4 Document Root and Path Structure

All software packages for the php implementation were installed in the MSProject/software directory. The document root for both Apache and PHP were set to the MSProject/webroot directory. The MySQL database schema was created under the MySQL directory, in a folder called data.

Under web root, the source code for the php classes defined was placed in the classes folder. (The actual class defined is defined in section 5). The common directory held files / pages that were included across multiple pages, or were files that contained utility methods. The help and images folders contain the help and image files for this project, while the login folder contains the pages that handles login and logout functionalities.

The following was the structure used in the PHP implementation:

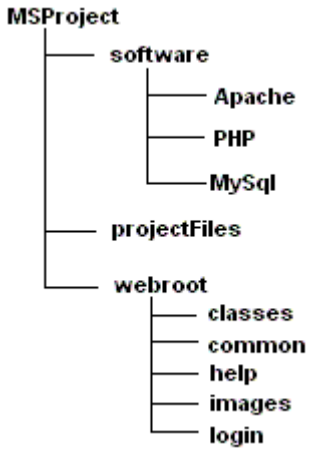

Figure 3: PHP implementation document tree

The projectFiles folder is used to store the files for each project. A separate directory is created programmatically for each project, and the files uploaded for that project are placed in the folder. The folder was placed outside the web root, as the intention was to have the same project file repository for both the PHP and the .NET versions. The path to this repository was added to the configuration file so the PHP pages could reference this location easily. Under Unix, the apache user account (or the user under which apache executes) should have permissions to the projectFiles folder. This can be achieved by either creating this folder as the apache user, or by giving write permissions to this folder to a group, and adding the apache account to the group.

4.2.5 Life Cycle of a Web Request in Apache/PHP/MySQL platform

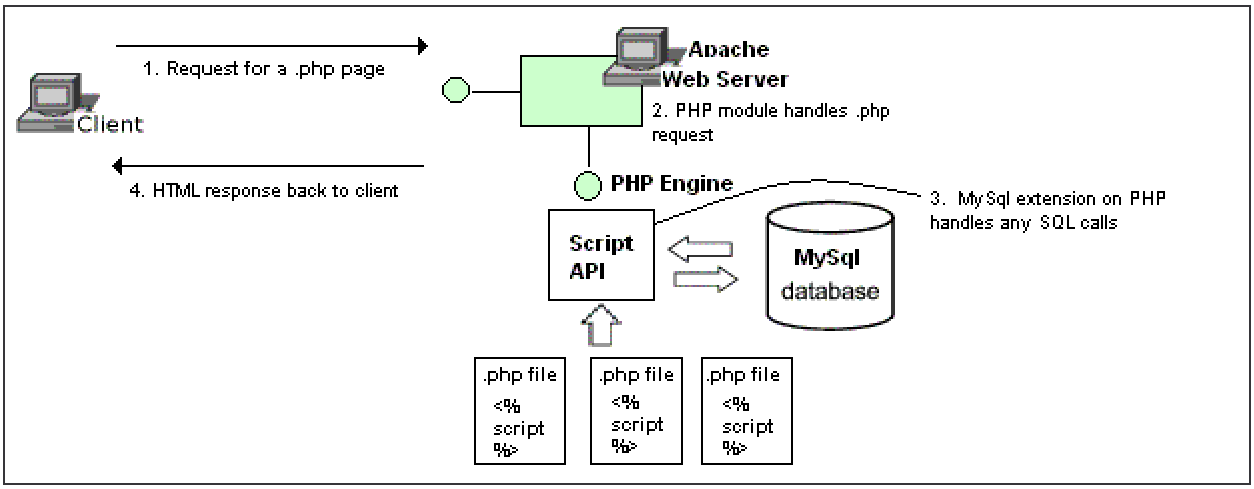

Figure 4: PHP request life cycle

1. User clicks on a link to a .php page on the browser.
2. The web server receives the request. The "LoadModule", "AddModule" and "AddType" directives instruct Apache to delegate the request to the PHP module.
3. The PHP engine receives the request, interprets the php code enclosed within the tags.
    a. Support for ODBC and MySQL is built in the PHP engine, and any MySQL calls are sent to the MySQL database engine through the MySQL driver.
    b. The MySQL engine executes the SQL and returns the results. The MySQL driver does appropriate data type conversions.
4. The PHP engine completes the translation of PHP code to HTML and sends the result back to the client browser.

4.3 ASP.NET Design

The ASP.NET implementation uses .NET web services and ASP.NET web forms. The .NET web services communicate with the MySQL database (the same schema as defined above). While the developed services and ASP.NET pages can be deployed on one or more IIS servers, the focus of this project is to concentrate on the development of the pages and not on the deployment. The latest version of Visual Studio .NET ("Whidbey") was used as the Integrated Development Environment, which has a web server built in. The Web Services and the ASP.NET pages are both served by the built in web server, but on different ports.

4.3.1 Visual Studio .NET Installation

The "Whidbey" edition of Visual Studio is still in the "alpha" testing stage. Dr. Axel Schreiner obtained the software during the Microsoft Professional Developer's Conference (PDC). This release of Visual Studio and the .NET Framework offers innovations and enhancements to the class libraries, common language runtime (CLR), programming languages, and the integrated development environment (IDE). Since "Whidbey" is self-contained, there is no additional configuration to be done after the software has been installed. The setting up of the projects for Web Services and Web Forms is described in section 4.3.3 and 4.3.4.

Whidbey supports the latest industry standards, such as Web Services Enhancements (WSE) libraries. WSE facilitates creation of business-critical Web services that involve multiple trust domains, long-running operations, Direct Internet Mail Exchange (DIME) and peer-to-peer communication. As WSE gains broad adoption and industry specifications for advanced Web service development mature, the WSE libraries will be integrated into the .NET Framework. Future releases of WSE will contain the latest version of these specifications and will include a policy framework, enhanced security model, and a SOAP messaging infrastructure. In addition, WSE will support hosting outside of IIS (as an executable or a Windows Service) and will support both HTTP and TCP communication.

The Visual Web Developer in "Whidbey" supports multiple websites - File System web sites, local IIS web sites, Frontpage / Sharepoint websites and FTP websites. The Web developer web server runs locally, specifically build to serve ASP.NET web pages. It runs as the current user, and does not require a separate IIS user account or ASPNET user account. It can be configured to use windows integrated security for access to file, network and database resources. The Visual Web Developer was used to serve the ASP.NET pages and web services, but on different ports.

4.3.2 Document Root and Path Structure

Since there are two parts to this implementation, there are two "document roots" defined. The DotNetServiceRoot acts as the starting point for the Web Service. The project uses "CodeBehind" files as opposed to "inline" files. In "CodeBehind" files, the web service/UI declarations are separated from logic, each defined in their own files. This will be explained in more detail in the next section. The Web Service files are stored directly in the document root, while the C# files that contain the actual code are in the code directory. Under this directory, folder structures corresponding to the package hierarchy used are created.

The following document structure is used for the ASP.NET implementation:

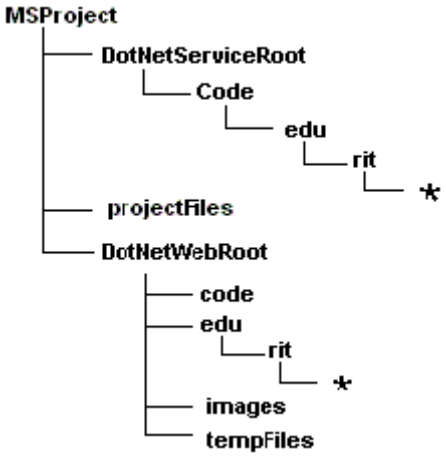

Figure 5: ASP.NET implementation document tree

For example, a C# file for the ProjectHandler - edu.rit.MSProject.Services.Projects.ProjectHandler was created under the code/edu/rit/MSProject/Services/Projects folder, and the file was named as ProjectHandler.asmx.cs

The project files folder is the same as the one defined for PHP - so that both versions of the project can access the same set of files.

The ASP.NET client files are contained in the DotNetWebRoot folder. Again, using code-behind techniques, the web files are placed directly under the document root, while the C# files are under the edu directory, based on the package structure. The Code directory contains the Proxy files for connecting to the web service. More information on the proxy file generation is explained in section 4.3.5.3. The images are stored in images folder, and the tempFiles folder is a temporary folder that contains the project File currently being previewed, as explained in section 5.2.2.2 - ManageFiles.aspx.

4.3.3 ASP.NET Web Services Setup

To create a new .NET Web Services Web site,

- Select File → New → WebSite. Under Project Types, select **General (C#)** and **ASP.NET Web Service** under templates. Set the location to be the MSProject/DotNetServiceRoot.
- Select File → Add New Item. Select **General (C#)** the category, and **Web Service using Code Separation** as the template. Set the name of the web service.
- This creates an .asmx file and an .asmx.cs file in the DotNetServiceRoot directory. In order not to clutter this directory, create the folders corresponding to the package structure, and change the location of the .asmx.cs file to the appropriate folder, and change the reference in the .asmx file.

4.3.3.1 Code Separation

XML Web services consist of two parts: the XML Web service entry point and the code that implements the XML Web service functionality. In ASP.NET, the .asmx file is a text file that serves as the addressable entry point for the XML Web service. It references code in pre-compiled assemblies, a code-behind file, or code contained in the .asmx file itself. The WebService processing directive at the top of the .asmx file determines where to find the implementation of the XML Web service.

4.3.3.2 WebService Processing Directive

At the top of the .asmx page is a WebService processing directive, which includes information in the form of attributes regarding the implementation of the XML Web service. For example, <%@ webservice **language**="C#" **codebehind** = "~/Code/edu/rit/MSProject/Services/Projects/ProjectHandler.asmx.cs" **class** = "edu.rit.MSProject.Services.Projects.ProjectHandler" %>

The **Language** attribute indicates the programming language used to develop the XML Web service. XML Web services can be created in any .NET-compatible language, such as Visual Basic .NET or Visual C#. The **Class** attribute indicates which class in the code-behind file implements the functionality of the XML Web service. This project uses C# as the implementation language.

4.3.3.3 Web Service Base Class

The System.Web.Services.WebService class defines the optional base class for XML Web services and provides direct access to common ASP.NET objects, such as those for application and session state. The XML Web service can inherit from this class to gain access to ASP.NET's intrinsic objects, such as Request and Session. Creating Web Service files using the template automatically use this class as the base class.

4.3.3.4 WebService Attributes

Each XML Web service requires a unique namespace, which makes it possible for client applications to differentiate among XML Web services that might use the same method name. For this project, the namespace was set to http://edu.rit/webservices/<servicename>.

4.3.3.5 Webmethod Attributes

To expose a method as part of an XML Web service, a **WebMethod** attribute must be placed before the declaration of each public method that needs to be exposed. Private methods cannot serve as the entry point for an XML Web service although they can be in the same class and the XML Web service code can call them. The following Web Method attributes can be set: **BufferResponse**, defaults to true, indicates if the response is buffered; **CacheDuration** to indicate in seconds how long the response can be cached. (Setting this to 0 disables caching). **Description** describes the web method; **EnableSession** determines if the web service is to participate in a session; **MessageName** allows to uniquely identify the web method, which comes in handy in cases where overloaded methods are used. If not, the MessageName defaults to the method name. Finally, a **TransactionOption** of Disabled, NotSupported, Supported, Requires or RequiresNew controls if the web service participates in a transaction.

4.3.3.6 *Life Cycle of a SOAP request in the .NET World*

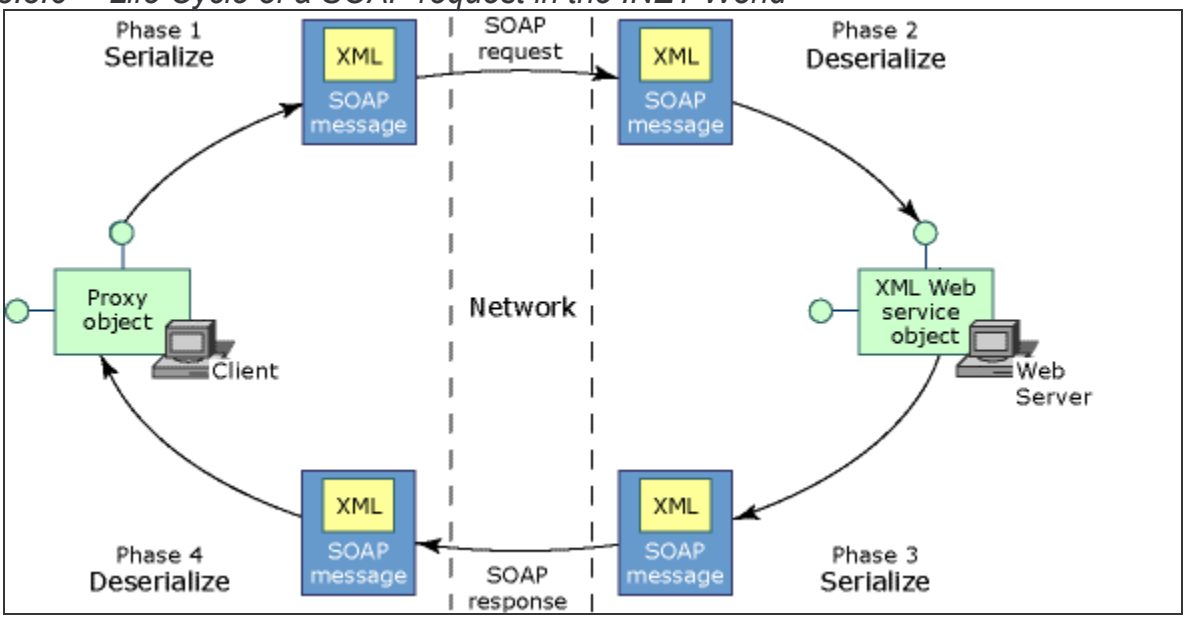

Figure 6: Soap request life cycle

The following describes the sequence of events that occur when an XML Web service is called:

1. The client creates a new instance of an XML Web service proxy class. This object resides on the same computer as the client.
2. The client invokes a method on the proxy class.
3. The infrastructure on the client computer serializes the arguments of the XML Web service method into a SOAP message and sends it over the network to the XML Web service.
4. The infrastructure receives the SOAP message and deserializes the XML. It creates an instance of the class implementing the XML Web service and invokes the XML Web service method, passing in the deserialized XML as arguments.
5. The XML Web service method executes its code, eventually setting the return value and any out parameters.
6. The infrastructure on the Web server serializes the return value and out parameters into a SOAP message and sends it over the network back to the client.
7. The XML Web service infrastructure, on the client computer, receives the SOAP message, deserializes the XML into the return value and any out parameters, and passes them to the instance of the proxy class.
8. The client receives the return value and any out parameters.

4.3.4 ODBC Connection Setup

Since MySQL is ODBC 3.5.1 compliant, the ODBC 3.5.1 driver that ships with MySQL can be used to connect to MySQL from ASP.NET. ASP.NET controls and code rely on data sources being created through the control panel. To do this on Windows XP, go to Control Panel → Administrative Tools → ODBC → Data Source Administration →MySQL 3.51 driver. Create the data source using user account created for the .NET project (.NetUser) The OdbcConnection provider for .NET provides for connection pooling.

4.3.5 ASP.NET Web Forms Site

To create a new .NET Web Services Web site,

- Select File → New → Website. Under Project Types, select **General (C#)** and **ASP.NET Web Site** under templates. Set the location to be the MSProject/DotNetWebRoot.
- Select File → Add New Item. Select **General (C#)** the category, and **Master Page using Code Separation** OR **Content Page using Code Separation** as the template. Set the name of the web page.
- This creates an .aspx file and an .aspx.cs file in the DotNetWebRoot directory. In order not to clutter this directory, create the folders corresponding to the package structure, and change the location of the .aspx.cs file to the appropriate folder, and change the reference in the .aspx file.

4.3.5.1 Master Pages

Master Pages is a new feature of ASP.NET "Whidbey" that allow for the creation of a consistent layout for all pages in an application. A single master page defines the look and feel and standard behavior for all the pages (or a group of pages). Individual content pages that contain the content you want to display can then be created. When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page. A Master page is a .NET file with the .master extension. They can be defined at the site, folder or page levels, and can contain almost anything that can be included in a normal asp.net page.

When ASP.NET receives a request for a content page, it performs the following steps:

- Fetches the page.
- Checks whether the content page references a master page.
- If so, fetches the master page associated with the content page.
- Merges the contents into the content placeholders on the master page.
- Renders the result to the browser.

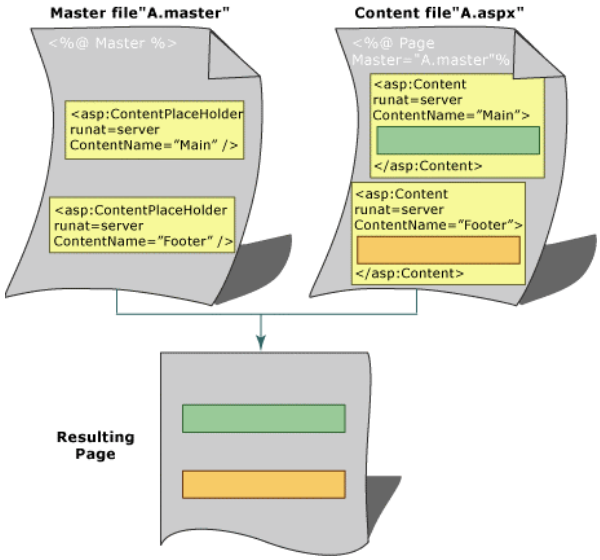

Figure 7: Master Pages Compile-time behavior

4.3.5.2 Code Behind Classes

Visual Web Developer can also create pages in which the HTML and control declarations are stored in an .aspx file but the code for the page is stored in a separate file. Separating the declarative elements and code can make it more practical to divide the development between a designer and a programmer. Similar to the definition of asmx files, aspx files can contain the following definition:
<%@ Page language="C#" compilewith="~/edu/rit/MSProject/Web/ProjectDetails.aspx.cs" classname="edu.rit.MSProject.Web.ProjectDetails_aspx" %>

Pages that use code separation take advantage of a feature referred to as partial classes. At runtime, the compiler uses the reference in the @Page directive to find the file containing the code. It then compiles the .aspx page and code page into a single assembly for execution.

4.3.5.3 Proxy Classes

In this project, since we are both the provider and consumer of XML Web service, we know the location and function of the XML Web service. The Add Web Reference dialog box can be used to locate and generate the proxy class. For a given address, the dialog box interrogates the Web site using an algorithm designed to locate XML Web service discovery (DISCO) documents and ultimately, the Web Services Description Language (WSDL) document. After having located an XML Web service, clicking the Add Reference button instructs "Whidbey" to download a copy of the service description to the local machine and then generate a proxy class for accessing the chosen XML Web service. The proxy class contains methods for calling each exposed XML Web service method both synchronously and asynchronously.

4.3.5.4 Dynamic and Static URLs

A Web reference can use either a static URL or a dynamic URL. The Web Reference URL property of the Web reference is used to specify the URL of the referenced XML Web service. By default, this property is set to the URL of the XML Web service selected, which is a static URL. If the URL Behavior property is set to the default value of **Static**, the proxy class's URL property is set using a hard-coded URL when an instance of that class is created.
If the URL Behavior property of the Web reference is set to **Dynamic**, the application obtains the URL at run time from the <appSettings> section of the application's configuration file, such as:

```
<appSettings>
  <add key="MSSProject.Services.ProjectHandler"
       value="http://edu.rit/webservices/ProjectHandler.asmx/>
</appSettings>
```

When an instance of a proxy object is created, the URL property of the object can also be set programmatically. Regardless of which URL the proxy uses, it must be for an XML Web service that conforms to a WSDL that matches the one used when adding the Web reference. Otherwise, the proxy class that was generated earlier will not be able to interface with it.

4.3.5.5 WSDL.EXE

An alternative for using the Add Web Reference dialog is creating the proxy class using the wsdl.exe tool. This is the tool used by Visual Studio internally to create the proxy class when adding a Web reference. This is useful in cases where we are unable to access the XML Web

service from the machine on which Visual Studio is installed, such as when the XML Web service is located on a network that will not be accessible to the client until run time. The generated file can then be manually added to the project. This technique was used in this project, and the generated files were placed under the code directory.

The proxy class generated by the Add Web Reference process derives from the **System.Web.Service.Protocols.SoapHttpClientProtocol** class, which contains several properties like ClientCertificates, CookieContainer, Credentials, Proxy, Timeout and Url that can be used to control the behavior of how this class accesses an XML Web service. For DIME support however, the base class has to be changed. The reasons for this will be explained in a subsequent section.

5 Implementation Specification

This section describes how the two websites were implemented. The use cases specified in the system analysis broadly categorize how the existing website was implemented, and form the basis of the two implementations. This section references the use cases, and for each use case, describes the implementation details for each technology used. A single use case was sometimes implemented using multiple pages, and is described as such in separate subsections.

5.1 Browse Projects Use Cases

The Browse Projects use cases can be broadly broken into the following steps: Browse for projects by Year, Browse by Advisor, Browse by Student, Search for projects and View Project details. Sub-section 5.1.1 describes the PHP implementation of these five pages, whereas 5.1.2 describes the .NET implementation details.

5.1.1 PHP Implementation of Browse Projects

PHP allows for the definition of classes and objects. While not truly object oriented, PHP classes allow for a collection of related variables and functions. Objects can be created using the "new" keyword, and discarded using the "unset" statement.

5.1.1.1 Class Definitions

For the Browse Projects functionality, the following classes were defined:

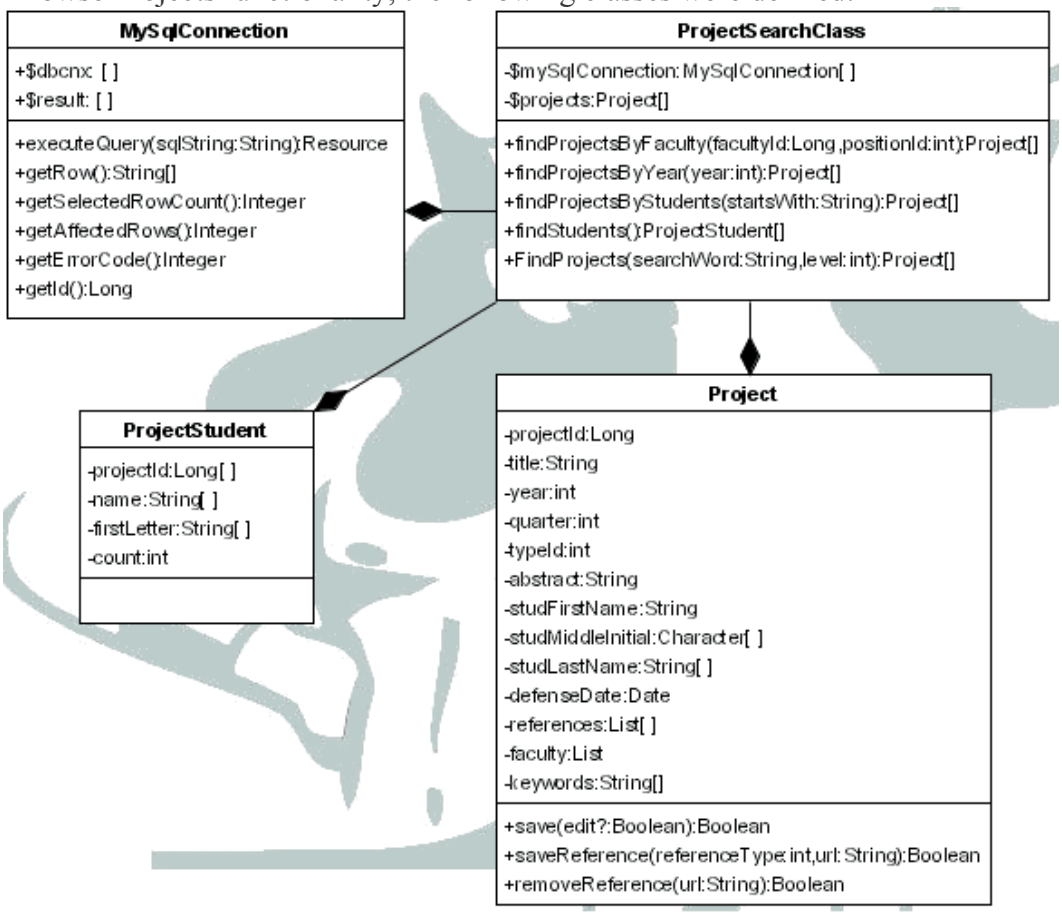

Figure 8: PHP Classes to support Browse functionality

MySqlConnection:

This class abstracts out the MySql Connection. MySql is bundled with PHP by default. This is done by compiling PHP with the --with-mysql [=DIR] directive. The DIR references the directory where MySQL is installed. Once installed, the MySQL extension allows for MySQL specific commands.

- When a new object is instantiated, it establishes a connection to the MySQL database using the following commands:

Resource **mysql_connect** ([string server [, string username [, string password [, bool new_link [, int client_flags]]]]])
Which returns a MySQL link identifier on success, or `FALSE` on failure.

Mysql_connect () establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *`server`* = 'localhost: 3306', *`username`* = name of the user that owns the server process and *`password`* = empty password. If a second call is made to mysql_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The *`new_link`* parameter modifies this behavior and makes mysql_connect() always open a new link, even if mysql_connect() was called before with the same parameters. The *`client_flags`* parameter can be a combination of the constants MYSQL_CLIENT_COMPRESS (to use compression protocol), MYSQL_CLIENT_IGNORE_SPACE (allows space after function names) or MYSQL_CLIENT_INTERACTIVE (use interactive_timeout instead of wait_timeout). The MySqlConnection class does not use any client flags, and sets the new_link value to false.

bool **mysql_select_db** (string database_name [, resource link_identifier])
Returns **`TRUE`** on success or **`FALSE`** on failure.
mysql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mysql_connect** was called without arguments, and use it.

- Once a connection is established, subsequent calls to execute queries from this object use the same connection. Queries are executed through the executeQuery() MySQLConnection command, which uses:

resource **mysql_query** (string query [, resource link_identifier])

mysql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if mysql_connect() was called with no arguments, and use it. The result of the query is buffered.
mysql_query() will also fail and return `FALSE` if you don't have permission to access the table(s) referenced by the query.

- Assuming the query succeeds, the getAffectedRows() or getSelectedRows() can be called, which internally call **mysql_num_rows()** to find out how many rows were returned for a SELECT statement or **mysql_affected_rows()** to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

Only for SELECT, SHOW, DESCRIBE or EXPLAIN statements, mysql_query() returns a new result identifier that can be passed to other functions dealing with result tables.

- Once a select statement has been executed, the result set can be iterated through the getRow() function, which abstracts out the call to:

array **mysql_fetch_array** (resource result [, int result_type])
Returns an array that corresponds to the fetched row, or **`FALSE`** if there are no more rows. The result_type is an enumeration repsenting if the array should be numeric (accessible only by the column index), associated (only by column name) or both. If two or more columns of the result have the same field names, the last column will take precedence in the case of the associative type. To access the other column(s) of the same name, the numeric index of the column or an alias for the column must be used.

ProjectSearch Class:

The project search class has the following methods:

- FindProjectsByFaculty(facultyId, positionId)
  This method takes the facultyId as a mandatory parameter, and the positionId as an optional parameter. All projects advised by the given faculty in the given position are queried, and a resultset returned. The positionId is an enumeration to signify the roles of Chairman, Reader or Observer. If no positionId is passed, all roles are considered. This method returns an array of Project Objects.
- FindProjectsByYear(year)
  This method takes the year as a parameter, and returns all the projects that were started in that year. Returns an array of Project Objects.
- FindProjectsByStudents(startsWith)
  For the given alphabet (startsWith), this method queries the student table for all students whose last names start with that alphabet, and returns the projects undertaken by those students.
- FindStudents()
  Returns a collection of ProjectStudent objects. There will be 1 projectstudent object for each alphabet, with a count of the number of students whose last names start with that alphabet (0 if none). If there is only one student whose last name starts with that alphabet, that student's name is returned, along with the projectId for the project the student worked on.
- FindProjects(searchWord, level)
  For the given search word, queries the database to find all instances where the word occurs. The tables to search are determined by the level, which could be a combination of Project Title, Keyword, Student name and Faculty Name. Based on the search criteria, a list of projects is returned.

Project Class:

This class represents the state of a Project. For the BrowseProjects use case, only the state of the object is queried, acting more like a value object. The behavior of this object is used only in the ManageProject use case. The methods this object provides are:

- SaveProject(action)
  This method persists the project information to the database. Based on the action, either new (in case the details are for a new project) or edit (when a project is being edited), the information is either inserted or updated in the database. This also involves creating/editing project keywords, creating/editing project faculty map records and storing student information.
- SaveReference(referenceType, url)
  This method saves the url in the ProjectReferences table. The reference Type can be an enumeration representing Project Proposal, Project Defense, Reference Links or other downloads. The url is a string referencing the absolute url in case of reference links, and a relative url for the others. The final url in this case will be constructed by adding the Project File location (obtained from the configuration settings), the project Id and the value in the url field.
- RemoveReference(url)
  This method removes the url saved in the ProjectReferences table.

ProjectStudent Class:

This class is a value object used only by the FindStudents method. This class only has state, and no behaviour. The information stored by any object of this class is the count of students who have undertaken projects, and whose last names start with the same first letter, the student name and project id in case there is only one student.

5.1.1.2 Common PHP Pages

MasterQueries.php

This page contains a set of utility functions. Including this page in other pages gives access to the functions defined in this page. Since PHP does not support the concept of static functions, this common page with pre-defined functions serves that purpose. The functions exposed in this page are:

- FindCommitteePositions
- FindReferenceTypes
- FindProjectTypes
- FindAllFacutly - this function returns only faculty members who are involved in projects under various capacities.

5.1.1.3 UI Design

The "Listing.php" page consists of three parts - the left navigation frame, comprising of the list of faculty and the years in which projects were undertaken, the right navigation frame, consisting of student names, and the center frame, which has 4 tabs, with the "Browse" tab highlighted. The center tab displays the project details based on the option/link selected from one of the navigation frames.

Figure 9: index page

The left navigation frame, "browse.php" contains two tabs - the "Browse by Advisor" and "Browse by Year" tab. These tabs are created by using the common functions to find all faculty members, and all years in which projects were done. The faculty members and years are linked to the "Listing.php" page, passing in the appropriate facultyId or year. The ProjectSearch class' methods will be used to retrieve the projects advised by the passed in faculty, or the projects undertaken in that year.

The right navigation frame, "students.php" uses the ProjectSearch class to retrieve a list of students who have undertaken projects. The frame lists all letters of the English alphabet, and if there exists one or more students whose last name starts with for the given letter, then the letter is hyperlinked. If there is only one student whose name starts with that letter, the link directly takes the user to the project details page. If multiple students exist, the user is taken to a second-level browse page, where the student names are displayed with hyperlinks to the project details page.

The two navigation frames are linked to the Listing.php page using IFRAMEs.

Search.php

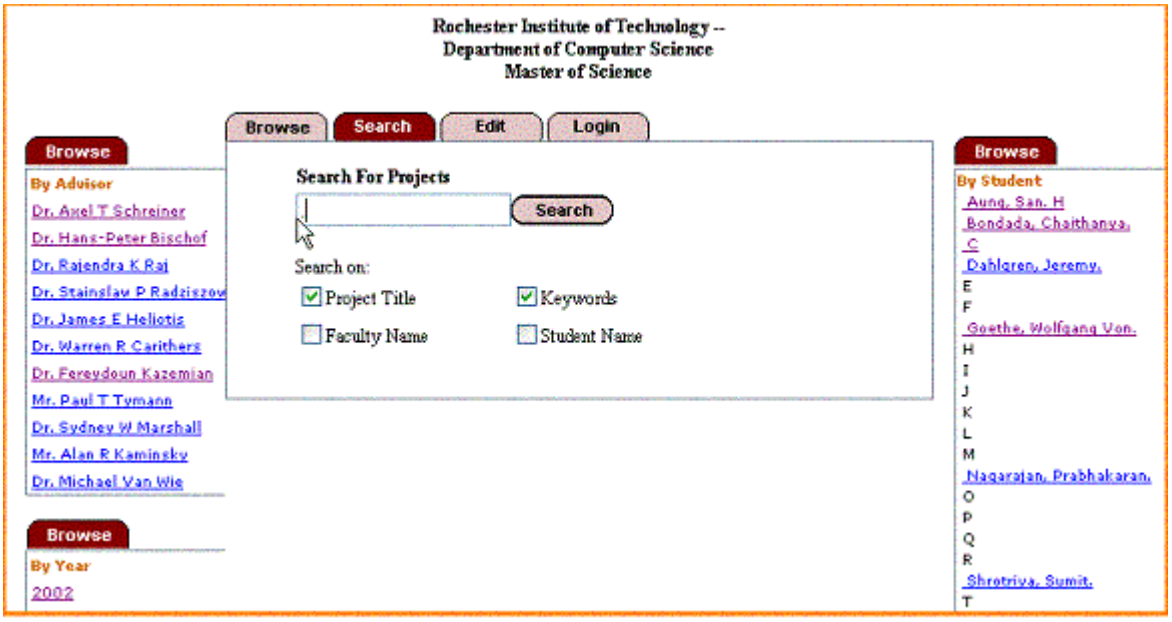

Figure 10: search page

The "Search.php" allows for users to search for a project based on words founds on the title, keywords, faculty name or student name. Based on the search criteria, the FindProjects method of the ProjectSearch class is used to retrieve project details.

Validation: The search term should not be empty, and either one of the four criteria selection checkboxes must be selected.

5.1.2 ASP.NET Implementation

The ASP.NET implementation of the Browse Project use case is implemented in two parts: The Web Service portion and the ASP.NET web client. This is explained in more detail in the following sub-sections.

5.1.2.1 ASP.NET Web Service Implementation

5.1.2.1.1 Services Defined

"**BaseService**" is a base Service class, which extends System.Web.Services.WebService, is defined with the intention that all Web Services defined for this project will extend from this class. This class contains a single method to raise an exception via a SoapException. The "RaiseException" method has the following signature:

public SoapException RaiseException(string uri, string webServiceNamespace,
string errorMessage, string errorNumber, string errorSource, FaultCode code)

Where **uri** represents the actual web service where the exception occurred (will be a sub-class of the base class); **namespace** where the web service occurred (which will be of the form

http://edu.rit/webservices/<webservice name>); **errorMessage** is a detailed description of what the error is; **errorNumber** signifies a code associated with the message, **errorSource** is where the error occurred (which might be the database, network etc. based on the actual error); **code** is an enumeration for whether it was a client error (data sent by the client was invalid) or a server error (a server constraint).

The "Master" or reference lists are defined in the **MasterList** Web Service, which extends the BaseService class. The idea behind this web service is to provide a list of reference items that the Web client can use to display - as a list of options to select from, instead of hard coding in the web page. Any data that is not dependent on a particular project can be retrieved from this web service. To support this, the following operations are defined:

- CommitteePositions()
  This web service returns a list of committee positions - currently, Chairman, Reader and Observer, and the Ids corresponding to these positions.
- ReferenceTypes()
  This web service returns a list of reference types- currently, "Project Proposal", "Project Report", "Reference Links" and "Other downloads", and the Ids corresponding to these types.
- ProjectTypes()
  This web service returns a list of project types - currently, Thesis and Project, and the Ids corresponding to these types.
- ProjectYears()
  This web service returns a list of years where projects are undertaken. This will most likely contain all years from when the website goes live (all years where data has been entered).
- FacultyInfo()
  This web service returns a list of Computer Science faculty currently in the database. The faculty information return includes the Faculty name and their Id as stored in the database.
- ProjectsByStudents()
  This web service returns a list of the count of projects done by students, sorted by the student's last name. Unlike the PHP implementation where there was an object returned for each letter, the ASP.NET implementation returns only objects where there exists at least one student whose last name begins with that letter.

"ProjectHandler" extends the BaseService class, and is a service that retrieves project details. The following operations pertain to the Browse Project use case:

- ProjectsByYear(int year)
  This method takes the year as a parameter, and returns all the projects that were started in that year. Returns the ProjectCollection object, which is a collection of ProjectInfo objects. The WSDL generator converts this to an array of a complex datatype called ProjectInfo.
- ProjectsByStudents(String startsWith)
  For the given letter (startsWith), this method queries the student table for all students whose last names start with that letter, and returns the projects undertaken by those students. Returns a ProjectCollection object.

- ProjectsByFaculty(long facultyId)
  This method takes the facultyId as a parameter. All projects advised by the given faculty in all capacities are queried, and a ProjectCollection returned.
- ProjectsByFaculty(long facultyId, int position)
  This is an overloaded method that takes the facultyId and the positionId as parameters. The positionId is an enumeration to signify the roles of Chairman, Reader or Observer. All projects advised by the given faculty in the given position are queried, and a ProjectCollection returned. Since this is a web service, the webMethod name should be unique - overloaded methods cannot be distinguished based on their parameters. The WebMethod has an optional attribute called MessageName that can be used in these situations to define a unique name for the service. For this web method, the message name is defined as: MessageName="ProjectsByFacultyAndPosition".
- FindProjects(String searchWord, int level)
  For the given search word, queries the database to find all instances where the word occurs. The tables to search are determined by the level, which could be a combination of Project Title, Keyword, Student name and Faculty Name. Based on the search criteria, a ProjectCollection object is returned.

5.1.2.1.2 Classes Defined:

The following classes were defined to support the above web services:

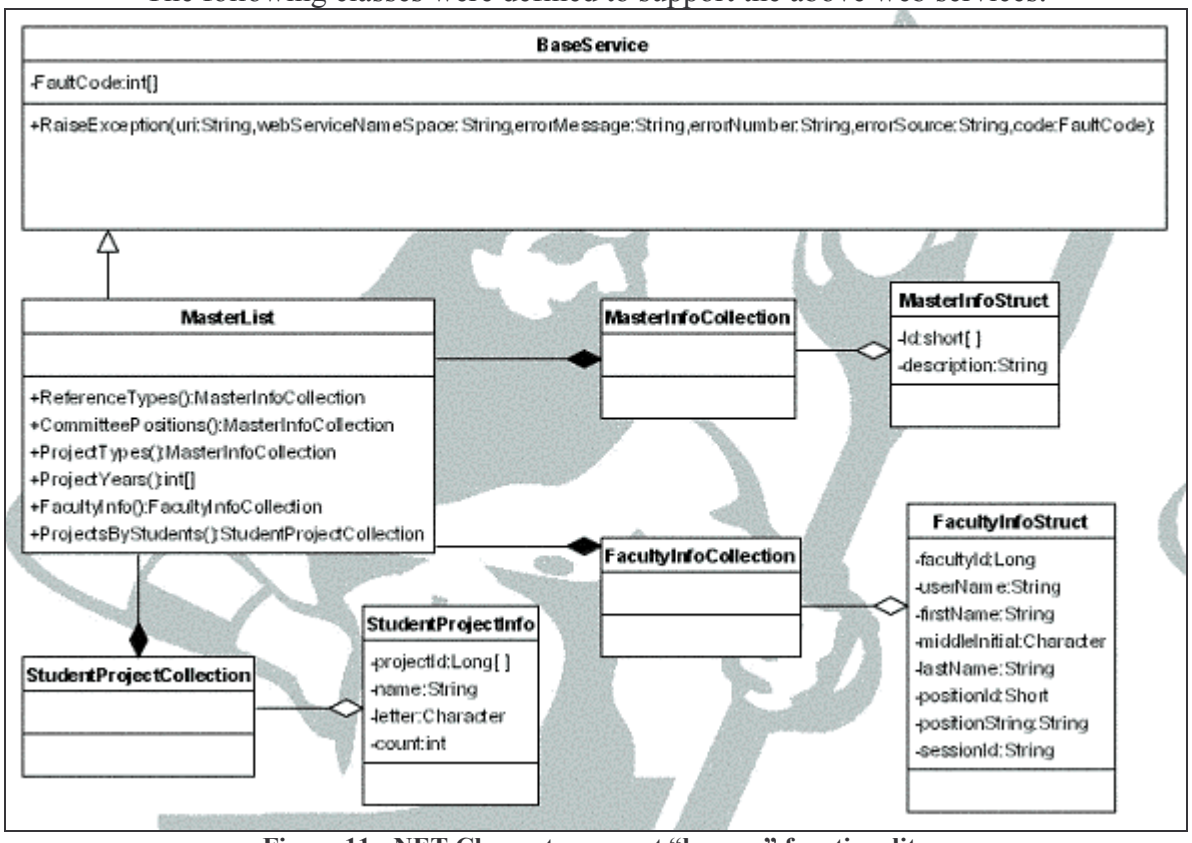

Figure 11: .NET Classes to support "browse" functionality

The following classes were defined as "value objects" - to pass data back and forth between the web service and the client. These classes do not have any behavior; they only store data.

- FacultyInfoStruct
- MasterInfoStruct
- StudentProjectInfo

Since these classes are to be used as return values for a web service invocation, these classes should be serializable - they must define a no-argument constructor in addition to any other constructors defined. Moreover, the WSDL is generated automatically based on the publicly scoped variables for the class. Hence all the variables that need to be available to the client are defined as public variables.

There are 3 collection classes, as suggested by their naming convention, which extend the Systems.Collections.CollectionBase class. The reason for extending this class and creating a collections class is to ensure type safety. If a collection class that can hold any object were used, the client does not know what kind of object to expect. Moreover, strongly typed collection classes lead to more descriptive WSDL files, as Visual Studio internally transforms them into arrays of the objects they hold. "Whidbey" also introduces the concept of "generics". Generics allow programmers to write generic code without specifying a data type. Generics force the compiler to produce intermediate language code that accepts the data type specified by the consumer of the class. While generics are a powerful concept, this project does not make use of generics. The collection classes are:

- FacultyInfoCollection
- MasterInfoCollection
- StudentProjectInfoCollection

5.1.2.2 ASP.NET Web Site Implementation

The ASP.NET web site implementation relies heavily on the Master Pages concept to design the layout of the pages. The PHP site uses IFRAMES to achieve this. ASP.NET pages can be constructed using either server controls or regular html controls. Server controls have many advantages over regular html controls including:

- The ability to generate events that result in post-backs to the server. Any event that normally is handled in the client can also be handled in the server, without any additional code to determine if the page has been submitted.
- The ability to dynamically create controls and add them to a page. Setting certain properties can change the behavior of a control.
- Server controls are correctly rendered based on the requesting browser. For example, Netscape does not support the isEnabled property on checkboxes. If a regular html checkbox control were used, additional code needs to be placed to simulate a "disabled" (grayed out) control. Server controls automatically take care of the rendering based on the browser.
- Advanced widgets are available that are not available as html controls. Examples include the loginControl and TreeControl.

The Master page defines “Content Placeholder” panels that are templates, and should be overridden by the child pages. The child Content pages can place “Content” panels in the placeholder panels, and controls can either be statically or dynamically added to these panels. The Master page for this project defines 4 content placeholders, one for the each browse tab (browse by advisor, browse by year and browse by student) as well as one placeholder for the listing area. The Master page also defines the content tab for the three “browse” areas, but having content placeholders gives flexibility for the content page extending the Master page - the individual content pages can choose to override the controls available in the Master page, and replace them with their own controls.

The three “Browse” tabs - searchContent, yearContent and studentContent are dynamically generated on the “PageLoad” event of the Master Page. The MasterInfo web service is invoked to get the faculty list, valid years and the student list. As explained earlier, the wsdl.exe tool was used to generate the proxy class.

The “Listing.aspx” file is the content page that inherits from the MasterPage, and provides the listing of projects based on the selected criteria. On an initial page load, the current year's projects are displayed. On subsequent loads, when the user clicks on either the faculty name, year or student name, the appropriate parameter is passed to the listing page. If the faculty name is clicked, the facultyId is sent as a parameter to this page; if the year is clicked, the value is passed; and if a letter in the studentContent tab is clicked, the selected letter is passed as a parameter to the listing page. The query string is obtained by examining the request, and based on the parameter passed, the appropriate web service is invoked and the result is displayed.

The UI is very similar to the page design of the PHP site. The index page of the ASP.NET site looks exactly like the PHP site index page, as depicted in Figure 9: index page

5.2 Manage Projects Use Cases

A pre-condition of the "Manage Projects" use cases are that the faculty member should actually be logged in. In addition to providing the login functionality, the Manage Projects use cases allow for the faculty to search for projects, edit a project, create a new project and edit the project files. The following sub-sections describe in greater detail the implementation intricacies for both technologies. 5.2.1 talks about the PHP implementation, while 5.2.2 details the .NET implementation.

5.2.1 PHP Implementation of Manage Projects

5.2.1.1 Class definitions

A new class is defined specifically for handling login-related functions. The "LoginHelper" class has the following functions defined:

- function Login($user,$pass)

  This function takes in an username and password, uses the MySQLConnection class to validate against the MySQL database. In case the username and password are valid, a PHP session is created, and the username, facultyId and the role of the user are stored as session variables. The return of this method is a boolean indicating whether the login was successful.
- function isLogin()

  Utility method that interrogates the session to see if there exists a username and facultyId in the session. Availability of these in the session indicates that the user has logged in. Since the session is per user, and is file based, there is no risk of hijacking another user's session.
- function Logout()

  Removes the objects that login inserted into the session, and explicitly destroys the session. This ensures that even if the same sessionId is regenerated later, users cannot mistakenly be logged in as someone else.
- function getLostPass($email)

  Accepts and email address, and uses this to search the database to check if a username exists that is associated with the passed in email id. If there exists one, invokes the SMTP server to send an email to the address on record, with the username and password.

5.2.1.2 Common Functions

Since the Manage Projects deals with session variables, and inserting and deleting objects from the session, a common function to retrieve and place objects in the session was defined. Multiple pages display the same option boxes (to select/display the quarter, year, faculty) and display the project details in a tabular format, so common functions were defined for handling these tasks.

- retrieveFromSession(varName)

  PHP supports instantiating an object given the implementing class name, similar to the classloader functionality in Java. Given the classname as the parameter to the function, a new object of this class can be created by simply invoking a new on the variable name! This method checks to see if an object exists in the session that implements this class, if so, returns it. If not, instantiates one, and returns it. This ensures that only a single instance of the class exists in the user's session.
- writeYearHtml(name, selectedYear)

Generates html code for selecting a year. If the optional parameter of selectedYear is passed, the SELECTED option is set for that year.

- writeQuarterHtml(name, selectedValue)

  Similar to writeYear, this generates the html for selecting a quarter.

- writeFacultyHtml(name, facultyId, isEditable)

  Generates the html to select the faculty name - can be used for selecting the Reader or Observer. An optional parameter - isEditable decides if the return should be html code for a dropdown selection, or a label with a hidden field containing the facultyId.

- printResults(resultSet, loginHelper)

  This method generates the project list in a tabular format. The loginHelper class is used to determine if the user is logged in, and if so, compares the username with the position of the project and checks the user's role to determine if the edit button should be available.

5.2.1.3 UI Design

Login.php

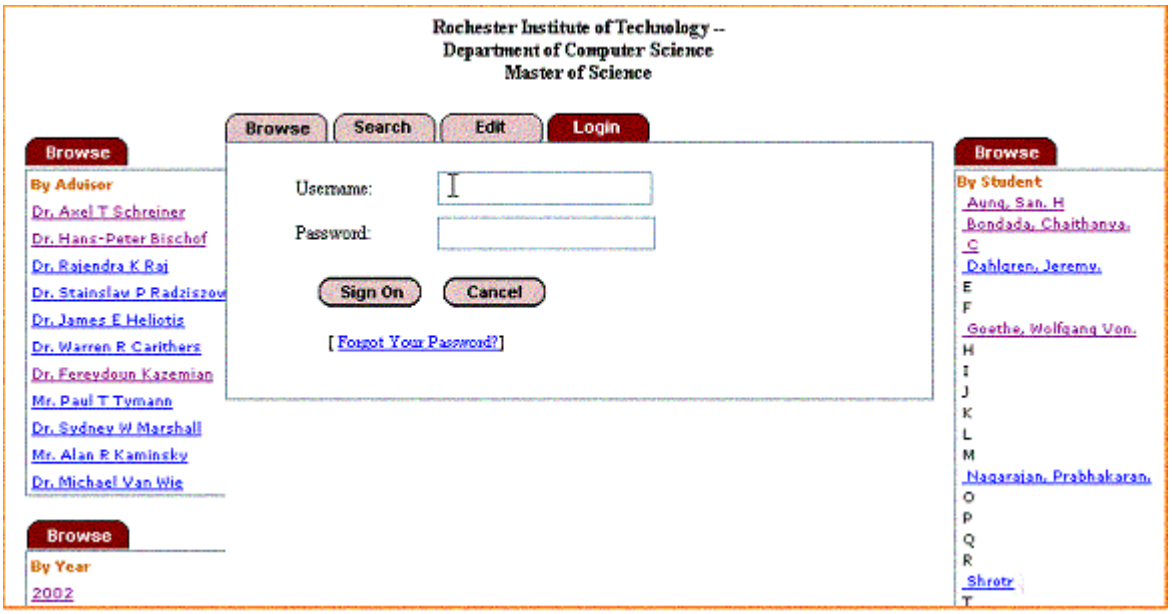

Figure 12: Login page

The login.php page includes the two browse pages using IFRAMES just like the listing.php page. The login page allows the faculty member to enter their username and password. Clicking on "Sign On" after entering the username and password validates the entered information and takes the user back to the listing page. A new session is created and the username, facultyId and role information are stored in the user's session. The role information is used in conjunction with the project details to determine if the edit button/link is available next to the project. For users with the "Advisor" role, the project's chairman id is compared to the faculty Id from the session, and if they match, the edit link is displayed next to the project. If the role is that of "Graduate Coordinator", the edit link is displayed on all projects.

Validation:
Client Side: Both the username and password should have values filled in before the "Sign On" button can be clicked.
Server Side: The username and password are validated against the faculty table for validity. If they don't match, an error message is displayed to the user. If authenticated successfully, the user is taken to the listing.php page.

The "Forgot Your Password" link can be used to retrieve lost passwords. Invoking this link brings up a page where the user's email address can be entered. The entered email is validated against the faculty table, and if the given email address is found, the username and password are sent to that address.

EditProject.php

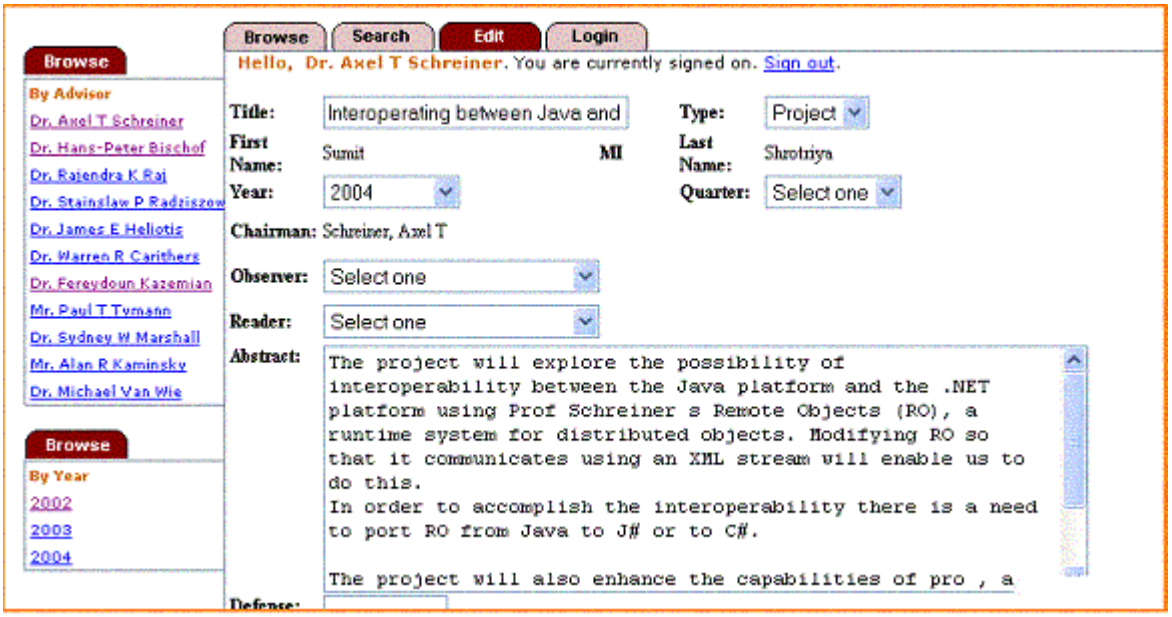

Figure 13: Edit Project page

The editProject.php page is used for both creating and editing projects. A parameter passed ("action") to this page determines the mode in which this page operates. If the mode is edit, implying that a project detail is to be edited, the projectId must be passed as a parameter to this page. In the "create" or "new" mode, a new Project object is created with null values even before the page is displayed. This helps in populating the form with the project attributes.

In the edit mode, the projectId is used to retrieve the project details from the database. The ProjectSearchClass has a getProjectDetails method that takes in the projectId as a parameter. This method queries the database, populates the project object with the values from the database, and returns it. The project Chairman, and student names cannot be edited, and hence are displayed as labels instead of an editable box.
In the create mode, all the values are empty except for Chairman, which defaults to the faculty logged in, and cannot be edited.

Once the required information is entered, the project information can be saved using the “save” button, or the “cancel” button used to cancel all changes (retrieves the information again from the database, thereby invalidating all changes made). There also exists a “Manage Files” button that allows for project files to be uploaded to the server. In order for files to be uploaded successfully, the project information has to be saved (a projectId is required for the upload files to work successfully. The reason for this is explained in the Manage Files section). Clicking on the “Manage Files” before saving the project prompts the user to save the project before attempting to upload files.

Validation

Client Side - Edit Mode: The title, Type, Year and Quarter must be filled in before the project can be saved.

Client Side - Create Mode: In addition to the above, the student first name and last name have to be entered before the project can be saved. The project has to be saved before the manage files button can be invoked.

ManageFiles.php

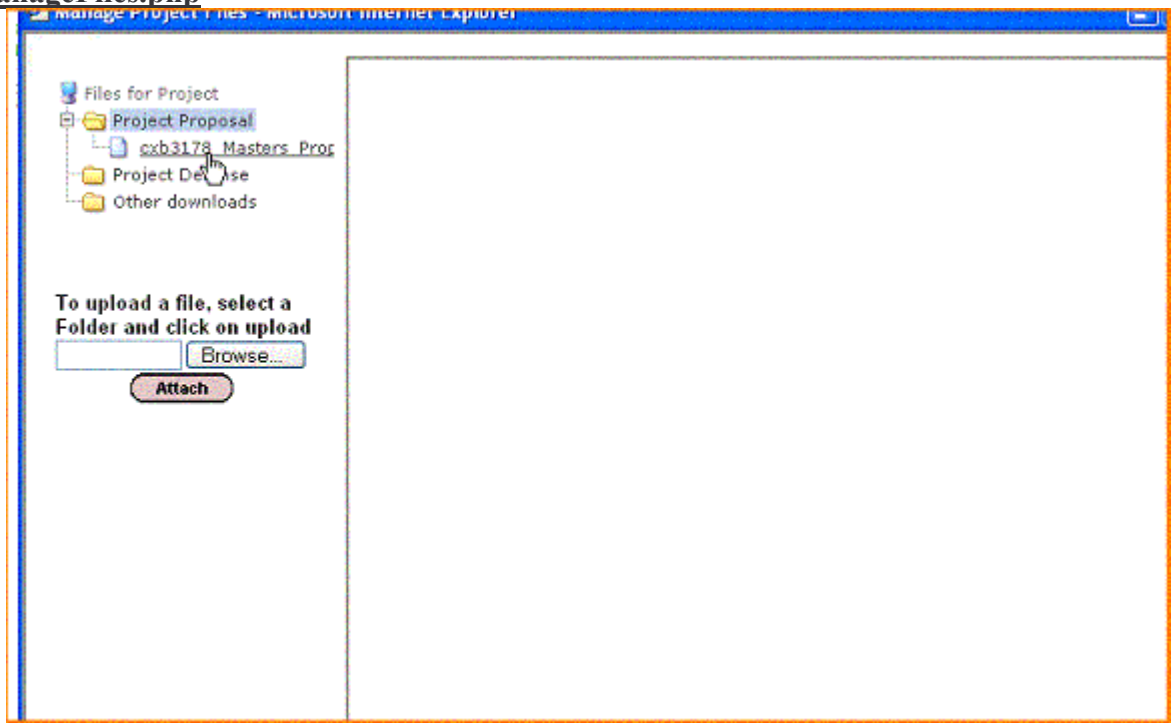

Figure 14: Manage Files page

The Manage Files section, as the name suggests, allows the faculty member to upload files for the project. Currently, there are 3 sections defined under which files can be uploaded - Project Report, Project Proposal and Other downloads. The files are uploaded using standard html input control - <INPUT TYPE=”FILE” ...>. The directory to which files are uploaded is set outside the document root (that is accessible to both the PHP implementation and the .NET implementation). The location of this document root is specified in a constants file for easy modification. Within this directory, sub directories are created for each project. This eliminates a clutter of files existing in the same directory, as well as accidental overwriting of another project's file with the same name - it is conceivable that multiple students name ther project

proposal as “Masters proposal.doc”. In situations like this, if each project had its own directory, name collisions will not occur. Since we need a directory per project, a directory is created with the project Id the first time an upload occurs (the project title cannot be used, as this is likely to change).

The Manage Files page uses a tree structure for displaying the folders and existing files under that folder. The three pre-defined categories appear as folders in this page. However, this is a UI construct, and the folders do not physically exist in the server. There is currently no provision for creating additional folders under the specified categories. When a file is uploaded, the category needs to be selected first (the selected “folder” is highlighted), and the file uploaded under that category. This causes two actions: the file is uploaded to the server, and an entry made to the ProjectReferenceMap table indicating the filename and the category under which it was uploaded. This is used to recreate the tree structure for future edits. (Please note that the tree structure is created from the database, and not by inspecting the server file system. This is faster, but there exists a possibility that if the file is physically deleted from the server file system, the file name may still show up in this structure)

The “Browse” button can be used to browse the client file system and select a file to upload. Once a file is selected, the “Attach” button can be used to upload the file to the server. Upload occurs only if a “folder” to upload to has been selected. After the file has been uploaded, link to view the file appears under the uploaded category, with the target set to the preview pane on the right. The link can be clicked on to verify that the correct file was uploaded.

Next to any uploaded file is also a link to delete the file. The “trash can” icon appears next to each file, and clicking the file causes the file to be deleted on the server, and the reference removed from the ProjectReferenceMap table. This can be used to delete older versions of files if the student submits a newer revision of a document with a different name or if the wrong file was uploaded.

Validation:
Client-Side : A category (or “folder”) must be selected before a file can be uploaded. The file input box must reference a valid file before the file can be uploaded.

5.2.2 .NET Implementation of Manage Projects

5.2.2.1 ASP.NET Web Service Implementation

5.2.2.1.1 Services Defined

In addition to the ones described in section 5.1.2.1.1, the following service was defined to handle retrieving and saving the project details:

“LoginHandler” extends BaseService and is used to authenticate the user. It defines the following two operations:

- Login(username, password)

The login web service takes two string parameters - the username and password, validates against the MySQL database, if valid, creates a sessionId, persists it in the database, and returns a FacultyInfoStruct object that has the sessionId populated. The sessionId is created by a combination of the username, a randomly generated string and the user's IP address. The idea behind the sessionId is that for subsequent requests, the client does not have to send the username and password to be authenticated; just the sessionId is sufficient. The sessionId is validated against the stored sessionId, and is used as the authentication token. The sessionId however, cannot be a simple string that can be guessed, hence the combination of some known values (username, ip address) and a randomly generated string.

- Logout (username)

Since web services in this project do not participate in sessions (even though ASP.NET supports session management for web services, this project does not use this functionality. The reason behind this is that the design accounts for the possibility that the web forms implementation might reside on a separate server than the web service, and in this scenario, web services cannot possibly participate in the same web forms session), the Logout function requires a username. Once the Logout function is invoked, the service deletes the sessionId from the database.

The following operations were added to "**ProjectHandler**" for the Manage Projects use cases.

- GetProjectDetails(projectId)

The GetProjectDetails web method takes a long value representing the proejctId as a parameter, and for the given projectId, queries the database to retrieve the project details. The faculty members, references and the keywords associated with the project are populated in the ProjectInfo object and returned.

- saveProject(ProjectInfo)

The ProjectInfo object is examined to see if the project id for this project exists. If the project id exists, this probably is an edit. This is confirmed by first selecting the count of projects with this id. If there is a record, that record is edited with the new values. If a record does not exist, a new record is inserted into the database. The map entries for the project are also saved.

- saveFile(projectId, referenceType, path)

This web service saves a file for the given project Id. The path parameter contains the file name; reference type the type (or category) of file; and the project Id is the project for which the file is to be saved for. Getting the project store from the configuration files, and adding the projectId to the file store create the full path. If a directory for the project does not exist, a new directory with the project id is created. The actual file to be saved is passed as an attachment to the web service, and hence does not appear in the parameter list. The attachments are uploaded using DIME format, which is explained in further detail in the next sub section.

- GetFile(projectId, path)

The GetFile returns the requested file as a DIME attachment. The full path to the requested file is obtained by getting the project file store location from the configuration file, and appending the projectId and the filename (path parameter). If the requested file does not exist, a null value is returned. If it does exist, the full path is returned to the caller in addition to the file being sent as an attachment.

- DeleteFile(projectId, path)

  Given the filename and projectId, this web service operation deletes the file from the file store. The return of this method is a boolean indicating if the operation was successful.

5.2.2.1.2 Classes Defined

As specified in the previous section, the LoginHandler, which extends the BaseService was the only new class defined for this use case. The ProjectHandler described in the previous section (Figure 11: .NET Classes to support "browse" functionality) was modified to implement new operations to save projects and project files.

5.2.2.1.3 DIME: Soap Messages with Attachments

This project uses the DIME format to upload/retrieve files when using web services. The Web Service Enhancements (WSE) pack provided as an extension to the .NET framework by Microsoft only supports DIME format and not MIME attachments. Before understanding how DIME was used, we first need to understand the need for DIME.

While one of the key strengths of SOAP is the ability to encapsulate XML data within a SOAP message, there are times when data does not come to us as XML. There are situations where the act of serializing data into XML will cause inefficiencies. For instance, image files come in a number of different formats but tend to be transferred across the Web in mostly JPEG or GIF formats. Both of these formats for holding image data is highly structured and could be converted into an XML schema. Nevertheless, images are large enough as it is, and the processing required to serialize the data to and from XML would involve a horrendous slow down to a mechanism that is well accepted and efficient.

There are also cases where legacy formats are used, for example systems using Electronic Data Interchange (EDI) formats. While it is certainly possible to convert the data to xml before sending it in a SOAP message, and converting it back to EDI at the receiving end, it will be much more efficient if the raw data was sent in the legacy format. DIME solves this issue by allowing attachments to be sent with SOAP messages. DIME is a specification for including multiple binary records within a single package. The records could contain any kind of data, including image and other binary files, SOAP messages (you could forward a SOAP request as an attachment), or even MIME messages. DIME is similar to MIME, but accounts for some of the inefficiencies of MIME multipart messages. With DIME, the data length information is included with the data header, as opposed to MIME where a boundary or separator string is provided at the beginning of a MIME message, and included between data records. The client has to parse through the data records to find this boundary, and only then is the data record accessible for processing. In addition to this, with MIME messages, some complicated logic has to be implemented to come up with a separator string that does not appear in the message itself.

Microsoft supports DIME attachments through its Web-Service Enhancements (WSE). The Microsoft, IBM and Verisign consortium proposed WSE to improve interoperability in areas that are crucial for Web services, such as security, reliable messaging, and sending attachments. Currently available as a Technology Preview, WSE is an add-on to Visual Studio.NET and the

Microsoft .NET Framework. After installation, the following section needs to be added to the Web.config file to support WSE:

```
<webServices>
  <soapExtensionTypes>
    <add type="Microsoft.Web.Services.WebServicesExtension,
Microsoft.Web.Services,
        Version=2.0.0.0,
      Culture=neutral,
      PublicKeyToken=31bf3856ad364e35" priority="1" group="0"/>
  </soapExtensionTypes>
</webServices>
```

For SOAP messages with attachments, the WSE runtime implements a DIME-compliant message parser that is able to translate the records of an incoming DIME message and extract the primary SOAP message from the first DIME record and any encapsulated files from successive records as attachment objects. After being extracted from DIME, the primary SOAP message is then passed to the WSE message pipeline where the series input filters evaluate the SOAP message for any other WSE-supported headers. The following figure shows how an incoming DIME message is handled by the WSE runtime and ASP.NET.

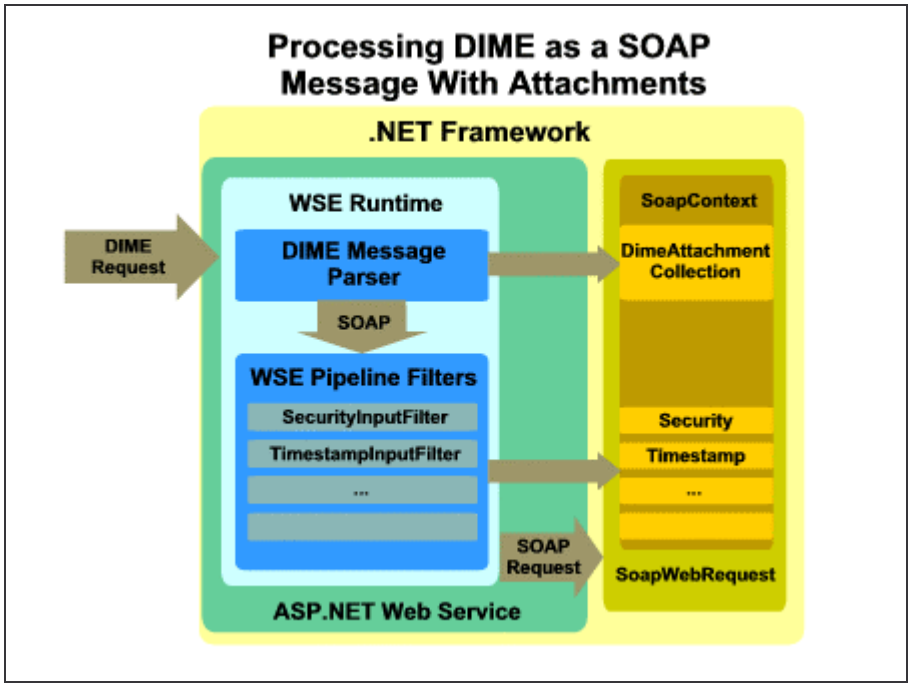

Figure 15: Dime Processing

When incoming DIME messages conform to the WS-Attachments specification, the WSE runtime handles them as SOAP messages with attachments, and then extracts the primary SOAP message and all of its attachments. For each attachment, a **DimeAttachment** object is added to the **DimeAttachmentCollection**, and these attachments are accessed through the **SoapContext.Attachments** property. Similarly, attachments can be added to the **DimeAttachmentCollection** for the **SoapContext** of an outgoing message so that the WSE runtime will include them as DIME records in the outgoing message.

For the client to successfully extract or attach DIME attachments, the proxy class must implement Microsoft.Web.Services.WebServicesClientProtocol instead of the standard System.Web.Services.Protocols.SoapHttpClientProtocol. The WebServicesClientProtocol class is defined in the WSE, and the assembly should be added to the ASP.NET client. The following section in the web.config file can do this:

```
<assemblies>
  <add assembly="Microsoft.Web.Services, Version=2.0.0.0,
     Culture=neutral,
     PublicKeyToken=31BF3856AD364E35"/>
</assemblies>
```

5.2.2.2 ASP.NET Web Site Implementation

The ASP.NET website implementation for manage projects follows along the same lines of the browse projects implementation - the implementation pages extend the same master page defined earlier. Similar to the PHP implementation, the faculty member must be logged in before they can create or edit any project details.

Login.aspx

The ASP.NET ”Whidbey” login controls encapsulate all the functionality required for securely managing users in a Web application. The login control eliminates the need to write any authentication and login code required for a web application, and enables easier registration of new users. It also includes functionality to authenticate users, change user passwords, and send password reminders.

Though this functionality was not used, it is worth mentioning that the LoginView control allows you to display different content templates to different users depending on the security roles they belong to. For example, certain sections of the web site can be allowed for users with a certain role, whereas advanced features may be made available for faculty admins. All this can be achieved by little or no programming effort. However, roles and rights that are provided by ASP.NET work only when users login, and work on individual pages, not on controls on a page. For this project, only faculty members login, and the functionality is not different based on who logs in. (Even though the graduate coordinator has slightly higher access than other faculty, the additional functionality does not translate into additional pages).

In addition to setting the display settings like background color and fonts for the text, the login control allows customized error messages to be displayed in case of a login failure. The control also allows recovering a lost password. During development, we can also specify if the “Remember me next time” option can be presented to the user. If this option is set, the user has the ability to automatically login every time they access the login.aspx page. This functionality is automatically implemented by ASP.NET by setting client side cookies.

There are various events that this control generates which can be used to control the behaviour of this control. For example, a “BeforeLogin”, “Authenticate” and “AfterLogin” triggers are available that can be used to plugin custom authentication methods. In this case, a web service call is made to validate the user account.

The page layout is similar to the PHP site layout, and is shown in Figure 12: Login page.

EditProject.aspx

The EditProject.aspx requires that the user be logged in. Once logged in, the functionality and validations provided by this implementation is exactly the same as the PHP implementation.

ASP.NET has a calendar control that is used in the implementation. The calendar control allows for selecting a date. The default date selected is the current date, and the control allows for navigating the calendar in monthly increments. The calendar control is rendered using JavaScript on the client browser. The calendar control is placed on a panel, and the arrow next to the defense date controls the visibility of the panel. Clicking on the arrow either displays the panel or hides it. The date for the control is set to either the date entered on the defense date field or the current date if the field is empty.

The page appears exactly like the PHP version (Figure 13: Edit Project page), but using ASP.NET's calendar control.

ManageFiles.aspx

The UI for the ManageFiles.aspx is similar to the PHP version. However, while the PHP implementation used JavaScript to create the tree structure, ASP.NET has a tree control available that was used for this purpose. The TreeView control provided by ASP.NET allows the control to be bound to a datasource, an xml file, or be dynamically constructed. This project dynamically constructs the nodes in the TreeView.

The treeview control is fully customizable by setting the images for the nodes, leaves and the connectors. The nodes can also have checkboxes that can be selected. Selecting the checkboxes generates events, and actions can be performed during these events on the server. However, in order to save the number of trips made to the server, the UI allows for multiple items to be selected, and then clicking on Delete to actually delete the files.

Similar to the PHP implementation, files can be uploaded by searching for the file from the file system (using the "File Upload" server control, and not the regular html control), and then clicking on the "Attach" button. The page appearance is shown in Figure 14: Manage Files page

On clicking the Attach button after selecting the file to upload, the file is sent to the server (ASP.NET web site server), which receives the file. The advantage of using the FileUpload server control is that posting the page sends the file to the server, but there is no complicated multipart/MIME-request parsing to extract the file. The FileUpload control on the server receives the file, and only when the saveFile method is invoked on the control is the file saved. In this project, all project information is handled by the web services, and the ASP.NET web site acts as the client, the web site server does not handle saving the file. Instead, this file is sent to the web service for persistence. On a post, the FileUpload.Content is used to get the IO stream. The IO stream from the file upload control is used to create a new DimeAttachment object, and this object is added to the SoapContext. The web service is now invoked, and the file is sent as an attachment to the web service, which takes care of storing the file in the file store.

Since the file is stored in the web service server, clicking on a node to preview the file must first download the file to the web site server. The getFile method in the web service retrieves the file,

and sends it down as an attachment. The web site server receives this file and temporarily stores it in a folder (which is set in the configuration file). The preview page displays this file when the link is invoked. The reason for storing the file temporarily is to capture the appropriate MIME type so that the browser can display the file. If the file were streamed directly from the web service web server, this could be achieved by redirecting the request. However, in this case, the MIME type has to be set programmatically. Instead of going through the hassle of maintaining a MIME type map that links a file extension to the MIME type, storing the file temporarily and allowing the browser to display it makes the process more efficient. The temporarily stored file is deleted the next time a preview of a different file is done by the user.

6 Comparison of technologies

As described in the project motivation earlier, one of the goals of this project is to compare the technologies involved. The existing implementation used Java servlets, and this project used PHP and ASP.NET as the technology choice. Since the same functionality was implemented using three different techniques, we can compare the three with respect to the overall architecture of the design, ease of learning and using the language, flexibility of the API, the response time of each implementation, as well as the tools available to develop and debug the implementation. In this section, for each aspect listed above, the strengths and weakness of each implementation choice is examined. The comparison is also tabulated as well as ranked in section 7.

6.1 Overall Design

6.1.1 Java Servlet Implementation

The existing implementation uses Java Servlet technology. The "admin" area has only one servlet doing the processing based on various actions to perform. The input xml is validated via dtds, and rendering is done via xslt processing. The "static" side is generated by periodic cron jobs that transform the xml to html.

Pros

- The overall design of this implementation is extremely simple.
- User Interface not very sophisticated - just enough to get the job done, which almost eliminates the UI design and implementation time. However, it must be noted that this is not really a fair comparison, as this implementation was done by Dr. Schreiner, and the UI would probably have been different were it to be implemented as part of this project.
- The static site page response time is extremely fast.

Cons

- There is only one servlet that does all the work. From a design perspective, there is no clear demarcation of roles and responsibilities. Having a single servlet do all the work results in a single-point of failure, and is generally not desirable.
- Need for a cron job to update the static site.

6.1.2 PHP Implementation

The PHP implementation uses PHP and MySQL to accomplish the same task. All the faculty information and "master" or reference table information is stored in the database to allow for future expansion.

Pros

- Fully database driven, any new faculty information or upload category can be added to the database and will immediately take effect without any change to the UI
- Richer UI than the Java Implementation.

Cons

- Page response time slightly slower than the Java implementation for the static site.

6.1.3 ASP.NET Implementation

The ASP.NET implementation consists of two parts - the ASP.NET web service, and the ASP.NET web client. The web service in turn talks to the MySQL database server.

Pros

- Similar to the PHP implementation, this design is also database driven
- UI comparable to the PHP implementation. Some advanced controls available to use, but were not used to keep the UI similar to the PHP implementation.
- There is a clear separation of the business tier with the UI/display tier.
- A new UI can easily be constructed on top of the web service layer, and this UI can be created using any other language as well.

Cons

- The design is more complicated than required. There are essentially two separate sites that could have been implemented using a single site.
- As a consequence of the design (two sites), performance is slow. This is not a drawback of the .NET framework, but is a result of the design approach taken.
- The project uses DIME as the protocol for uploading files as attachments. While DIME is one possibility, this protocol has not been ratified by W3C. There are competing protocols being considered including SOAP with attachments, as well as a competing protocol proposed by BEA.

Note: A lot of features used in this are based on the "alpha" release of Visual Studio .NET. There is a chance that some of the methods are deprecated by the time the product is released. Conversely, there could be more controls / features added by the official release.

6.2 Flexibility of API

This section compares the maturity of the language, as well as the API provided. Even though all the features provided by the language have not been tapped into by this project, the capabilities of the language will be considered. The availability of third party / open-source, readily available, plug-and-play capable packages each technology support will also be briefly mentioned here.

6.2.1 Java Servlet Implementation

The core of the java platform is the Java Virtual Machine (jvm) that interprets java bytecodes. A Java program is first compiled into java bytecode, and the jvm interprets this bytecode. Conceptually, Java is two things: the Java platform (runtime and APIs), and the Java language. The purpose of the Java platform is to support applications written in the Java language and compiled to java bytecode. Although there have been attempts to compile other languages to Java bytecode, these have largely been academic exercises. The ideal of Java has always been a *single language on multiple platforms*.

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets

can also access a library of HTTP-specific calls and receive all the benefits of the mature Java language, including portability, performance, reusability, and crash protection.

Today servlets are a popular choice for building interactive Web applications. Third-party servlet containers are available for Apache Web Server, Microsoft IIS, and others. Servlet containers are usually a component of Web and application servers, such as Apache Tomcat, BEA WebLogic Application Server, IBM WebSphere, Sun Java SystemWeb Server, Sun Java System Application Server, and others.

The first invocation of a servlet by a client loads the servlet into memory. Most servlet containers provide mechanisms for servlets to be loaded at startup to reduce the initial overhead. Once a servlet is loaded, it uses java's inherent multi threading capabilities to serve multiple concurrent requests. The servlet containers also contain mechanisms for database connection pooling, by which a pool of database connections can be established during startup, and reused as required.

Java, being inherently object-oriented, supports code reuse. Servlets are also classes that can inherit from other classes. Any user-defined servlet extends the HttpServlet, and the concept of interfaces can be used to define frameworks. Many open source frameworks exist that facilitate easy creation of xml configuration based web sites. Struts, a package developed by Apache, is an example of a web application framework based on the model 2 approach, a variation of the classical Model-View-Controller (MVC) design paradigm.

There also exist a plethora of tag libraries that use/extend servlets, JSP and JavaBeans to create personalized pages. Tiles, powered by Struts, is an example of a tag library that assists in creation of “portal” pages.

The Masters website implementation uses a limited set of features provided by the servlet specification. Since there is no database involved in the Servlet implementation, connection pooling is not used. The current design implements the entire functionality using a single servlet, eliminating the need for any framework classes.

6.2.2 PHP Implementation

At the heart of PHP 4.0 is the Zend Engine, an open source-scripting engine. PHP pages are compiled into opcodes, which are then interpreted by the zend engine. Because PHP is open source, it is supported by a number of different platforms and Web servers.

To an extent, PHP is object-oriented and it enables users to build classes and custom objects. However, for several reasons, PHP is not a true object-oriented programming environment. One of the most stunning examples of this is the scope of class member functions and properties. In PHP, all variables in a class are accessible externally for both reading and writing, making it impossible to hide a class' implementation. Additionally, the PHP language is not strongly typed, which causes problems when building larger applications, and makes debugging an application difficult. PHP allows for automatic variable declaration - i.e., if PHP encounters a

new variable at any point in the script, it will just instantiate the new one silently. This can cause numerous debugging problems.

Objects in PHP are language values in the sense that they are handled in much the same way as the simple object types (such as integer and string values). When performing operations like variable assignment and passing the object as a parameter to a function, the whole object is copied. Various reference “tricks” can be performed to simulate pass by reference, but the default language support is for pass by value.

PHP also misses several key elements present in most other modern development frameworks. One of the most important is structured exception handling. Though it is possible to register an error handler to process errors, the language itself provides no facility for this. There is no "try . . . catch" or a similar error / exception handler.

PHP has built in support for MySQL. However, this implies that the PHP language syntax for accessing MySQL database is very different from when accessing other databases. This makes it tedious to develop a web site using MySQL and porting it later to use a different database. PHP does not have any built-in multi-threading capabilities, but uses the one provided by the web server. Since PHP was designed specifically for web programming, it possesses some constructs that are commonly not available in other general-purpose languages. addSlashes() - to escape certain characters like ', “ etc; wordwrap(); html_entity_decode(); mime_content_type() are examples of web-specific functions.

Even though PHP has constructs to create objects and supports class definitions, PHP4 is not truly object-oriented. PHP does not support interfaces, and has no scope definition possibilities (all member functions and variables are implicitly public) PHP5 is being developed, and may be more object-oriented than PHP4.

PHP also has some popular open source implementations that can be customized. In addition to the standard “wiki” engines, complete content management systems like PHP Nuke, phpWebSite, and numerous PHP scripts are available for free download.

PHP provided just the necessary features required for this project. There were not a lot of additional features provided by PHP that was not used, and conversely, there was no functionality that could not be implemented because PHP did not support it.

6.2.3 ASP.NET Implementation

Between the three languages considered here, this is the lone “Microsoft only” platform. At the core of ASP.NET is the Microsoft .NET Framework. The framework provides the common language runtime (CLR) and the class library, upon which ASP.NET is built. When an ASP.NET resource is requested for the first time, the high-level programming code (such as Microsoft VB.NET or C#) is compiled to Microsoft Intermediate Language (MSIL) code. This code is then run by the CLR to generate machine code, which is then used to serve the page.

.NET is comprised of two things: the .NET Framework (runtime and APIs), and a plethora of supported programming languages. The purpose of the .NET Framework is to support applications written in any language and compiled to Microsoft Intermediate Language (MSIL). The goal of .NET is *a single platform shared by multiple languages.*
Microsoft currently provides .NET support for the following languages:

- C#
- Visual Basic .NET
- Managed C++
- JScript

Other vendors can come up with compilers to create MSIL code. This would make those languages .NET compliant. Currently, perl, python, COBOL and multiple other languages are being worked on to make them .NET languages. One of the key benefits of the framework supporting multiple languages is cross-language integration. Cross-language integration is the ability to seamlessly integrate multiple languages within the same application. As an example, you can write an application written in VB.NET, which can in turn call a C# class library.

ASP.NET forces developers to use a truly object-oriented approach to application design. All objects are true object-oriented objects supporting features such as inheritance, polymorphism (overloading of methods) and encapsulation. In addition, currently, all Microsoft supported programming languages in ASP.NET are strongly typed (including Jscript), similar to Java.

The .NET framework allows for fat-client applications (for windows based applications) and thin-client applications (for handheld devices) to be built, and the API has separate classes and methods for enabling this. This introduces additional complexity as the API has multiple classes with the same name, but present in different packages, each having a different target platform. Though some ASP.NET scripts and "tips and tricks" are available for download, it is rare to find a fully functional, feature rich ASP.NET implementation available for a free download. However, unlike PHP and Java, ASP.NET has built-in support for "portal" style pages. The new WebPartZone control in "Whidbey" enables clients to re-order Web content within their ASP.NET page, simply by dragging and dropping it within their client browser. ASP.NET then automatically remembers and stores these custom settings specified by visiting users within the new ASP.NET personalization service.

Even though the design of the ASP.NET implementation calls for the use of features (web-services) that are not used in the other implementations, this project still uses only a subset of features provided by the .NET framework. For the sake of keeping the UI the same between the PHP implementation and the .NET implementation similar, some of the features provided by the .NET framework were not used (like the multiview and wizard controls that allow for multiple pages of information to be captured and processed without having to write additional code to maintain session state). However, some advanced controls (login and treeView control) were used, which drastically reduced coding time when compared to the PHP implementation.

6.3 Tools (or IDE) Available

The availability of IDE and debugging tools influence to a large extent the popularity and the acceptance of a language. Readily available and intuitive debugging tools greatly help in

developing and debugging applications using the different languages. The three technologies being compared in this project do not require an IDE for page creation. Any text editor can be used to create the source files, and command line options used to compile them. This section briefly examines the tools available, and the reason for selecting the IDE used for this project.

6.3.1 Java

There are many Free and open source Java IDE available in the market today. The more popular ones are Eclipse, NetBeans, JEdit and JBuilder X Foundation. In the commercial market, even more powerful ones exist that integrate with the popular application servers available. For example, Visual Cafe and JBuilder integrate with BEA Weblogic, Websphere and Tomcat/JBoss. Almost all application servers allow for remote debugging as well - the IDE can "attach" to a running instance of a jvm and allow stepping through the code.

Java source files can also be created using any text editor, and compiled from the command line using javac, which ships with the java sdk (not the java runtime engine). With most text editors supporting Java syntax highlighting, this is in fact a popular choice for a lot of experienced programmers.

6.3.2 PHP

PHP support is also available in multiple IDEs. The most popular one is PHPEdit, which is freeware, and was the IDE used in this project. PHPCoder, HTML-Kit, PHP Expert Editor, SciTe are some of the other PHP freeware/shareware editors. Eclipse, the popular Java editor, has a plug in for PHP, and a lot of text editors have syntax coloring support for PHP. Some commercial editors like Zend Studio, PHP Ed exist for PHP as well.

PHPEdit has a built-in debugger, that acts as a web server and php pages can be debugged from within the editor itself. PHPEdit also has code insight, integrated PHP help and a help file generator (similar in concept to Javadocs). These are some of the reasons for using PHPEdit as the editor. The built-in debugger was the primary reason for selecting PHPEdit as the IDE in this project.

PHP authors also often use a regular text editor to author PHP pages. On invoking the PHP page from the web server causes the PHP engine to automatically compile the file if it detects some changes made to it between the source page and the compiled version.

6.3.3 ASP.NET

While the .NET Framework SDK is available for download from Microsoft, and the SDK allows for source codes to be compiled, proxy classes to be generated etc., very rarely do developers use the SDK without the accompanying IDE. Visual Studio is the proprietary Microsoft IDE that is used for ASP.NET development. For this project, the latest offering for Visual Studio .NET, codenamed "Whidbey" was used as the IDE.

"Whidbey" allows for debugging both web services and web pages through a built-in web server. For this project, since there were two separate sites defined, Whidbey automatically started two

web server instances, listening on different ports for the two servers. ASP.NET allows for debugging by setting two parameters in the page: debug=true or trace=true. This allows debug and trace statements to be left in the source even on production, but without the messages being printed (by just changing the Page parameter). Built-in debuggers allow for setting of breakpoints and stepping through the code. Visual Studio integrates with the Microsoft Software Developers Network (MSDN) library documentation to provide context sensitive help and access to technical articles and columns.

"Whidbey" also has "wizards" that automate creation of some classes, and has templates for web site creation. Even though this project did not use these advanced features, any beginner to the language will find these features extremely useful.

6.4 Ease of learning / using the technology

Even though there was no java implementation done for this project, this section comments on the ease of using it based on prior knowledge of the technology.

6.4.1 Java

Since there are numerous tutorials and sites available on the Internet, learning Java is relatively simple. However, the capabilities of Java are vast - for example, Java supports client applications through swing/awt classes, remote procedure calls using RMI, database support through JDBC, introspection and reflection classes, a new logging API, Java Server Pages and Servlets. Numerous other specifications exist like J2EE, an enterprise application specification, Java Data Objects JDO, Java Native Interface JNI, and Java Directory and Naming Interface JNDI. If the focus is one only servlets, learning and developing using servlets is straightforward, and can be easily accomplished.

6.4.2 PHP

PHP, as described earlier, was specifically designed for web applications. The idea behind PHP is to embed it in HTML pages, so the language, data types and functions it provides are very intuitive from a web development perspective. Additionally, with the plethora of web sites concentrating on PHP development, learning PHP is essentially a breeze.

6.4.3 ASP.NET

ASP.NET, unlike the other two languages described above, is more of a framework than a language. This framework supports multiple languages like C#, VB, C++ and JScript. Even when considering only a single language like C#, the features available are extensive. While not complicated to learn and develop in, only a subset of features was used for this project. Learning ASP.NET was easy mainly because of the usage of Visual Studio .NET. Visual Studio .NET allows for debugging from within itself (as it has a built in web server), and generates a sample web.config file with the required defaults. It also has a "design" view of the UI, where controls can be dragged and dropped, and the appearance of the controls during runtime can be established. The absence of this design view would have greatly increased the time to develop

the UI. If the same implementation were attempted using just the .NET framework classes, and using a different IDE, the progress would have been much slower.

6.5 Response Time for Each Implementation

Java and .NET employ both compilers and interpreters to process the source code, while PHP can be considered a fully interpreted language. The source file for Java and .NET is first compiled to some intermediate language (bytecode and MSIL respectively), and finally, when a browser/application requests the code, and interprets the compiled code and generates the final machine code to be executed. This section examines both the compile time and runtime performance of the technologies. Benchmark tests were not used to determine the compile and runtime performance, but as the pages were being compiled and tested, the time taken by the browser was noted. All background processes and applications were turned off when this test was performed to eliminate interference from other processes.

6.5.1 Java Servlet Implementation

The source files from Dr. Schreiner implementation were downloaded to my local machine to ensure that the target platform was the same in all cases, and compiled using the java sdk. The compile time of the code was fast. The runtime execution of the "admin" site was also good, with consistent 3-second response times. The "static" area was timed using the live site, and this was extremely fast as well - because there was no interpret/compile overhead. The static side was generated html, and the response time was only dependent on the web server performance.

6.5.2 PHP Implementation

The PHP pages compiled faster than the java classes. The page response time was definitely slower than the java implementation, but not very noticeable. The maximum time it took for a page to respond was 3 seconds, which includes the time to establish a connection to the MySql database. Performance on the PHP implementation can possibly be improved by using persistent database connections.

6.5.3 ASP.NET Implementation

The compile times using Visual Studio was much higher than the PHP or Java versions. This could possibly be attributed to the fact that Visual Studio introduced an overhead that was not present in the other implementations, as well as the fact that the two-site design called for more pages and services to be compiled. Moreover, the server controls used had to be converted to html code that works on both Netscape and IE, and this also contributed to the longer compile times. When the .NET framework SDK was used to generate the proxy classes (just for the sake of comparison), the generation was much faster than the corresponding time Visual Studio took to generate the web references, supporting the theory of Visual Studio introducing an overhead. The response time once the pages were compiled were comparable to the PHP implementation even though the ASP.NET did more work - the ASP.NET web server had to establish a connection to the web service through the proxy class, and then return the result from the service. It is quite possible that if the two components are installed in separate servers, the communication across the network could significantly increase the response times.

7 Summary

Comparison Criteria	Features		
	Java	PHP	.NET
Overall Design	+ Simple design - Single point of failure.	+ Flexible and richer UI - Slightly slower than the Java version.	+ Extremely flexible as well as robust - Slowest of the three implementations.
Flexibility of API	+ Extremely flexible. + Open source. + Available on multiple platforms. + Multiple database support through a single API. + Object-oriented language. - Still evolving. - Interpreted by JVM	+ Flexible + Open source + Available on multiple platforms. + Supports multiple databases - API database specific. - Procedural language - No Exception handling - Still Evolving - Interpreted by Zend.	+ Flexible + Supports multiple database through common API. + Object Oriented Language - Still Evolving - Not open source - Windows Based - Interpreted by CLR
Tools / IDE available	+ Multiple open source and free IDE + Open source Java Debuggers	+ Multiple free IDE available - No standalone debuggers	- No free IDE available - No standalone debuggers
Ease of Learning	+ Many tutorials freely available - Many specifications	+ Many tutorials freely available + Extremely easy to learn	- Not a lot of freely available material + Visual Studio drastically reduces learning time.
Response time	+ Extremely fast (for this design)	+ Acceptable response time	- Slowest response time (design choice)

Table K: Feature List Comparison

The following table summarizes the ranks for each implementation based on the criteria explained above. A rank of 1 signifies the best, and a value of 3 implies the lowest among the three.

Comparison Criteria	Ranking		
	Java	PHP	.NET
Overall Design Simplicity	1	2	3
Overall Design Flexibility	3	2	1
Flexibility of API	1	3	2
Tools / IDE available	1	2	3
Ease of Learning	2	1	3
Response time	2	1	3

Table L: Feature Summary

The Java Implementation scores on the simplicity of the design, but not on the flexibility as it relies on external systems to keep the data up to date, and the fact that a lot of information is built into the source in the form of constants (faculty list for example). The .NET implementation uses web services and an asp.net client, which segregates the business logic from the UI level, making it easy to construct a different UI on top of the web service. The UI can be created in any language desired, making this implementation the most flexible. PHP falls in between these two extremes. It uses databases to provide some flexibility, but is not as flexible as the .NET implementation.

While Java and .NET is equally feature rich, PHP provides an adequate set of functionality. During the implementation of this project, there was not an aspect that could not be implemented using PHP that could by implemented by the others.

In both the tools available and complexity of the language from a learning perspective, PHP trumps the other languages. Probably because PHP does not contain a lot of extraneous features like the other two, learning to develop applications using PHP is extremely easy.

Since there was no perceptible difference between the three implementations when measuring the response time, the three can almost be ranked the same. However, when factoring in the compile time as well, PHP can be ranked slightly higher than the other two

Based on this project's scope, the PHP implementation provides a compact set of features, is very easy to learn and develop, allows for a flexible design and generates an acceptable performance. PHP is ideally suited to develop applications where the load of concurrent users is not expected to be high, the user interface is not expected to be complicated, and the target audience is relatively fixed (not a lot of expectations for flexibility). The recommended implementation platform for this project is therefore PHP based.

8 References

- Apache, HTTP server project, http://httpd.apache.org/ABOUT_APACHE.html .
- Using Apache with Microsoft Windows, http://httpd.apache.org/docs/windows.html
- PHP, www.php.org
- MySQL, www.mysql.org
- PHPEdit, www.phpedit.net
- Wink, www.debugmode.com/wink
- Poseidon Community Edition, www.gentleware.com
- DIME: Sending Binary Data with your SOAP messages, http://msdn.microsoft.com/library/en-us/dnservice/html/service01152002.asp
- XML Web services created using ASP.NET, ms-help://MS.MSDNQTR.2003FEB.1033/cpguide/html/cpconASPNETImplementationOfWebServices.htm
- Web Forms, ms-help://MS.MSDNQTR.2003FEB.1033/dv_vbcon/html/vboriIntroToWebForms.htm
- Understanding Master pages, ms-help://MS.MSDNQTR.2003FEB.1033/dv_webnetcon/html/vbconUnderstandingMasterPages.htm
- JavaScript tree structure, www.destroydrop.com/javascript/tree